U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

NUREG/CR-6895, Vol. 2
ORNL/TM-2007/188, Vol. 2

Technical Review of On-Line Monitoring Techniques for Performance Assessment

Volume 2: Theoretical Issues

Office of Nuclear Regulatory Research

AVAILABILITY OF REFERENCE MATERIALS IN NRC PUBLICATIONS

NRC Reference Material

As of November 1999, you may electronically access NUREG-series publications and other NRC records at NRC's Public Electronic Reading Room at http://www.nrc.gov/reading-rm.html. Publicly released records include, to name a few, NUREG-series publications; *Federal Register* notices; applicant, licensee, and vendor documents and correspondence; NRC correspondence and internal memoranda; bulletins and information notices; inspection and investigative reports; licensee event reports; and Commission papers and their attachments.

NRC publications in the NUREG series, NRC regulations, and *Title 10*, Energy, in the Code of Federal Regulations may also be purchased from one of these two sources.

1. The Superintendent of Documents
   U.S. Government Printing Office Mail
   Stop SSOP
   Washington, DC 20402–0001
   Internet: bookstore.gpo.gov
   Telephone: 202-512-1800
   Fax: 202-512-2250
2. The National Technical Information Service
   Springfield, VA 22161–0002
   www.ntis.gov
   1–800–553–6847 or, locally, 703–605–6000

A single copy of each NRC draft report for comment is available free, to the extent of supply, upon written request as follows:
Address: Office of the Chief Information Officer, Reproduction and Distribution
Services Section
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
E-mail: DISTRIBUTION@nrc.gov
Facsimile: 301–415–2289

Some publications in the NUREG series that are posted at NRC's Web site address http://www.nrc.gov/reading-rm/doc-collections/nuregs are updated periodically and may differ from the last printed version. Although references to material found on a Web site bear the date the material was accessed, the material available on the date cited may subsequently be removed from the site.

Non-NRC Reference Material

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, and transactions, Federal Register notices, Federal and State legislation, and congressional reports. Such documents as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings may be purchased from their sponsoring organization.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at—

The NRC Technical Library
Two White Flint North
11545 Rockville Pike
Rockville, MD 20852–2738

These standards are available in the library for reference use by the public. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from—

American National Standards Institute
11 West 42nd Street
New York, NY 10036–8002
www.ansi.org

Legally binding regulatory requirements are stated only in laws; NRC regulations; licenses, including technical specifications; or orders, not in NUREG-series publications. The views expressed in contractor-prepared publications in this series are not necessarily those of the NRC.

The NUREG series comprises (1) technical and administrative reports and books prepared by the staff (NUREG–XXXX) or agency contractors (NUREG/CR-XXXX), (2) proceedings of conferences (NUREG/CP-XXXX), (3) reports resulting from international agreements (NUREG/IA–XXXX), (4) brochures (NUREG/BR–XXXX), and (5) compilations of legal decisions and orders of the Commission and Atomic and Safety Licensing Boards and of Directors' decisions under Section 2.206 of NRC's regulations (NUREG– 0750).

212–642–4900

U.S.NRC
United States Nuclear Regulatory Commission
Protecting People and the Environment

NUREG/CR-6895, Vol. 2
ORNL/TM-2007/188, Vol. 2

Technical Review of On-Line Monitoring Techniques for Performance Assessment

Volume 2: Theoretical Issues

Manuscript Completed: October 2007
Date Published: May 2008

Prepared by
J.W. Hines, D. Garvey, R. Seibert, and A. Usynin

The University of Tennessee-Knoxville
Department of Nuclear Engineering
Knoxville, TN 37966-2210

Oak Ridge National Laboratory
Managed by UT-Battelle LLC
Oak Ridge, TN 37831-6156

S.A. Arndt, NRC Project Manager

NRC Job Code N6080

Office of Nuclear Regulatory Research

Page intentionally blank

ABSTRACT

In 1995 the U.S. Nuclear Regulatory Commission (NRC) published a summary of the state-of-the-art for the area of on-line monitoring prepared by the Analysis and Measurement Services Corporation as NUREG/CR-6343, *On-Line Testing of Calibration of Process Instrumentation Channels in Nuclear Power Plants*. The conclusion of this report was that it is possible to monitor calibration drift of field sensor and associated signal electronics and determine performance of the instrument channels in a nonintrusive way.

In 1998, the Electric Power Research Institute (EPRI) submitted Topical Report (TR) 104965, *On-Line Monitoring of Instrument Channel Performance* for NRC review and approval. This report demonstrated a nonintrusive method for monitoring the performance of instrument channels and extending calibration intervals required by technical specifications (TS). A safety evaluation report (SER) was issued in 2000 in which NRC staff concluded that the generic concept of on-line monitoring (OLM) for tracking instrument performance as discussed in the topical report is acceptable. However, they also listed 14 requirements that must be addressed by plant-specific license amendments if the TS-required calibration frequency of safety-related instrumentation is to be relaxed. The SER did not review or endorse either of the two methods addressed in the topical report.

This report, published in three volumes, provides an overview of current technologies being applied in the United States for sensor calibration monitoring. *Volume 1—State-of-the-Art*, provides a general overview of current sensor calibration monitoring technologies and their uncertainty analysis, a review of the supporting information necessary for assessing these techniques, and a cross reference between the literature and the requirements listed in the SER. This volume entitled *Volume 2—Theoretical Issues* provides an independent evaluation of the application of the most commonly employed OLM methods. In empirical, model-based OLM, current measurements are applied to an algorithm that uses historical plant data to predict the plant's current operating parameter values. The deviation between the algorithm's predicted parameter values and the measured plant parameters is used to detect any instrument faults, including instrument drift. Many algorithms can be used to accomplish OLM; however, only auto-associative neural networks (AANN), auto-associative kernel regression (AAKR), and auto-associative multivariate state estimation technique (AAMSET) are investigated in this study. These techniques were chosen because they were either considered by EPRI's OLM working group that started in the 1990s, applied in EPRI's OLM Implementation Project which began in 2001, or are currently available as commercial products. *Volume 3—Limiting Case Studies* explores the inherent assumptions in the model whose impact on OLM is not yet known and also investigates other special limiting cases where the model behavior is unknown. The case studies reported in Volume 3 apply the modeling and uncertainty analysis techniques to a wide variety of plant data sets to consider the effects of these modeling assumptions and limitations.

The inclusion of a particular method should not be construed as an NRC endorsement for that technique. The theory behind each of these modeling techniques is explained in detail. The uncertainty of the model's prediction is also explained, as well as several methods for quantifying it. This expanded discussion on uncertainty is presented because OLM model uncertainty is one of the most critical issues surrounding the general acceptance of OLM as a valid technique for sensor performance assessment. If the uncertainty of the model is unknown, there can be no confidence in the model's predictions, making OLM for calibration extension pointless. Therefore, both analytical and Monte Carlo based uncertainty equations are developed for the three modeling techniques. Finally, the AANN, AAKR, and AAMSET models, along with their corresponding uncertainty values, are compared using both simulated and actual nuclear data. Although this comparison is theoretically thorough, it may not completely address the practical effects of the modeling assumptions and limitations. Therefore, Volume 3 of this series presents the results of applying the models and uncertainty analyses to a wide variety of plant data including those that violate some of the modeling assumptions; it recommends how these limiting test cases can be

identified, quantifies the effects of not meeting certain assumptions, and presents precautions to take to assure the assumptions are met.

Paperwork Reduction Act Statement

This NUREG does not contain information collection requirements and, therefore, is not subject to the requirements of the Paperwork Reduction Act of 1995 (44 U.S.C. 3501 et seq.).

Public Protection Notification

The NRC may not conduct or sponsor, and a person is not required to respond to, a request for information or an information collection requirement unless the requesting document displays a currently valid OMB control number.

FOREWORD

For the past two decades, the nuclear industry has attempted to move toward condition-based maintenance philosophies, using new technologies developed to ascertain the condition of plant equipment during operation. Consequently, in November 1995 the U.S. Nuclear Regulatory Commission (NRC) published NUREG/CR-6343, *On-Line Testing of Calibration of Process Instrumentation Channels in Nuclear Power Plants*, which summarized the state-of-the-art in the area of OLM. In that report, the NRC staff concluded that it is possible to monitor the calibration drift of field sensors and associated signal electronics, and determine performance of instrument channels in a nonintrusive way. Then, in 1998, the Electric Power Research Institute (EPRI) submitted Topical Report (TR) 104965, *On-Line Monitoring of Instrument Channel Performance*, for NRC review and approval. That report demonstrated a nonintrusive method for monitoring the performance of instrument channels and extending calibration intervals required by technical specifications (TS). The NRC subsequently issued a related safety evaluation report (SER), dated July 24, 2000, in which the staff concluded that the generic concept of OLM is acceptable for use in tracking instrument performance as discussed in EPRI TR-104965. However, the staff also listed 14 requirements that must be addressed in plant-specific license amendments if the NRC is to relax the TS-required calibration frequency for safety-related instrumentation. The SER neither reviewed nor endorsed either of the two methods addressed in the topical report.

This contractor-prepared NUREG-series report is the second volume of a three-volume set, which will provide an overview of current technologies being applied in the United States to monitor sensor calibration. Volume 1, published in January 2006, provided a general overview of current sensor calibration monitoring technologies and their uncertainty analysis, a review of the supporting information needed to assess those techniques, and a cross-reference between the literature and the requirements listed in the staff's SER. To augment that overview, this second volume presents an in-depth theoretical study and independent review of the most widely used online sensor calibration monitoring techniques, including the underlying theory and an evaluation of the inherent assumptions. The techniques reviewed in this volume were selected because they were considered by the EPRI OLM working group, were applied in the EPRI OLM Implementation Project, or are currently available as commercial products. By contrast, Volume 3, scheduled for publication by March 2008, will be dedicated to case studies that apply modeling and uncertainty analysis techniques to a wide variety of plant data sets to consider the effects of modeling assumptions and limitations.

The NRC staff anticipates that readers will use this reference to quickly locate the technical information required to assess the methods presented in plant-specific applications. This report is intended to provide the technical details that are necessary to conduct an accurate evaluation of each technique. This report should not be construed to imply that the NRC endorses any of the methods or technologies described herein; that is, a licensee would need to provide a complete description and justification for any proposed OLM approach.

Brian W. Sheron, Director
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission

Page intentionally blank

CONTENTS

Page

Page intentionally blank

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

Page intentionally blank

EXECUTIVE SUMMARY

For the past two decades, the nuclear industry has attempted to move toward condition-based maintenance philosophies using new technologies developed to ascertain the condition of plant equipment during operation. Specifically, techniques have been developed to monitor the condition of sensors and their associated instrument loops while a plant is operating. Historically, process instrumentation channels have been manually calibrated at each refueling outage. This strategy is not optimal in that sensor conditions are only checked periodically. Therefore, faulty sensors can continue to operate for periods up to the calibration frequency. Periodic maintenance strategies cause the unnecessary calibration of instruments that are operating correctly, which can result in premature aging, damaged equipment, plant downtime, and improper calibration under nonservice conditions. Recent studies have shown that less than 5% of process instrumentation being manually calibrated requires any correction at all [EPRI 2004]. Thus, plants are interested in monitoring sensor performance during operation and only manually calibrating the sensors that require correction.

In 1995, the Nuclear Regulatory Commission (NRC) published a summary of the state-of-the-art in the area of on-line monitoring; it was prepared by the Analysis and Measurement Services Corporation as NUREG/CR-6343, *On-Line Testing of Calibration of Process Instrumentation Channels in Nuclear Power Plants*. This report concluded that it is possible to monitor the calibration drift of field sensors and associated signal electronics and to determine performance of the instrument channels in a nonintrusive way.

In 1998, the Electric Power Research Institute (EPRI) submitted Topical Report TR-104965, *On-Line Monitoring of Instrument Channel Performance*, for NRC review and approval. This report demonstrated a nonintrusive method for monitoring the performance of instrument channels and extending calibration intervals required by technical specifications (TS). The calibration extension method requires an algorithm to estimate the process parameter. The NRC issued a safety evaluation report (SER) on TR-104965 dated July 24, 2000, concluding that the generic concept of on-line monitoring (OLM) for tracking instrument performance as discussed in the topical report was acceptable. However, the report listed 14 requirements that must be addressed by plant-specific license amendments if the TS-required calibration frequency of safety-related instrumentation were to be relaxed. Several of these requirements focused on the quantification of the uncertainty inherent in the process parameter predictions. This report provides a theoretical review of the empirical modeling techniques and the development and comparison of several methods for uncertainty quantification.

The goal of this three volume report, entitled *Technical Review of On-Line Monitoring Techniques for Performance Assessment*, is to provide the background and technical information necessary to assess the application of OLM techniques for sensor calibration extension. *Volume 1—State-of-the-Art* provides a general overview of the technologies being implemented for sensor calibration monitoring and the techniques used to quantify the uncertainty inherent in the empirical process variable predictions. It also provides a survey of the relevant information and a cross-reference between the relevant information and the 14 requirements. The survey provides a quick reference used to locate the technical information required to assess the methods expected to be presented in plant-specific license amendments. *Volume 2—Theoretical Issues* presents an in-depth theoretical study and independent review of the most widely used OLM techniques. It includes a presentation of the theory and an evaluation of the assumptions inherent in the empirical models and uncertainty quantification techniques. *Volume 3—Limiting Case Studies* investigates the effects of not meeting some of the modeling assumptions on model performance and uncertainty and gives guidance for identifying these situations.

As the second in the series, this report reviews several sensor monitoring techniques currently being considered by U.S. nuclear utilities for OLM applications. The methods evaluated in this report include auto-associative neural networks (AANN), auto-associative kernel regression (AAKR), and auto-

associative multivariate state estimation technique (AAMSET). Each of these methods is considered to be a plant-wide method, meaning that it calculates the parameter estimate using the plant historical data from a group of correlated, but not necessarily redundant, instrument channels. The group of instruments that constitute the model may come from any part of the plant, although they are typically grouped by a single system or subsystem. The theory and general application of these empirical methods are presented. Additionally, the report focuses on several methods used to quantify the uncertainties inherent in the model estimates. These uncertainty quantification methods include on-line analytical equations and off-line Monte Carlo-based techniques.

The basic components of empirical modeling uncertainty are derived, and a practical comparison between several of the techniques for estimating the uncertainty components is presented. In addition to describing the basic uncertainty components and how these can be quantified, the relationship between OLM and the nuclear plant's set point calculations, in terms of calibration uncertainty, is explained. Finally, the methods are applied to both actual and simulated nuclear plant data so that their results can be compared and validated.

ACRONYMS

AAKR	auto-associative kernel regression
AAMSET	auto-associative Multivariate State Estimation Technique
AANN	auto-associative neural network
ADVOLM	allowable deviation value for on-line monitoring
ALARA	as low as reasonably achievable
ANL	Argonne National Laboratory
ANN	Artificial Neural Network
CFR	*Code of Federal Regulations*
CI	confidence interval
CLT	Central Limit Theorem
CSA	Channel Statistical Allowance
CU	channel uncertainty
DP	differential pressure
EESE	Expert State Estimation Engine
EPRI	Electric Power Research Institute
EULM	error uncertainty limit monitoring
HRP	Halden Reactor Project
IANN	inferential artificial neural network
ICA	Independent Component Analysis
ICMP	Instrumentation and Calibration Monitoring Program
ICV	instrument calibration verification
KBH	thousands of pounds per hour (klbm/h)
M&TE	Measuring and Test Equipment
MAVD	maximum acceptable value of deviation
MSE	mean squared error
MSET	Multivariate State Estimation Technique
NLPCA	Nonlinear Principal Component Analysis
NLPLS	Nonlinear Partial Least Squares
NRC	Nuclear Regulatory Commission
NUREG/CR	NUREG prepared by a contractor
OLM	on-line monitoring
OLS	ordinary least squares
PCA	Principal Component Analysis
PE	Process Parameter Estimate
PEANO	process evaluation and analysis by neural operators
PI	prediction interval
PM	Process Measurement Effects
PSIA	pounds per square inch atmospheric
PSIG	pounds per square inch gauge
QA	quality assurance
RPSS	Reactor Parameter Signal Simulator
SCA	Sensor Calibration Accuracy
SD	Sensor Drift
SER	Safety Evaluation Report
SG	Steam Generator
SISO	single input single output
SMTE	Sensor Measurement and Test Equipment Accuracy
SPRT	sequential probability ration test

SPSS	Stochastic Parameter Simulation System
SRSS	square root sum of squares
SSE	sum of the squared error
TR	topical report
TS	technical specifications
TSP	Trip Set Point

1. INTRODUCTION

This volume is the second in the series of three. It focuses on the technologies currently being investigated for on-line instrument calibration monitoring (OLM) by utilities in the United States. On-line monitoring can be implemented for instrument monitoring, equipment monitoring, or operation monitoring; however, the term OLM is typically used to mean instrument calibration monitoring and will be used that way in this report. This volume explores the assumptions inherent in the technology and presents their impact on OLM. In particular, this volume investigates predictive uncertainty in detail. Quantifying the predictive uncertainty is one of the most challenging yet critical steps in gaining regulatory acceptance for OLM.

1.1 Background

For the past two decades, nuclear power utilities have been exploring technologies that determine the condition of plant equipment while the plant is operating. With these technologies, plants can implement more efficient condition-based maintenance strategies and replace some of the periodic maintenance tasks with a potentially more effective on-line method. Of special interest are the techniques that monitor the condition of sensors and their associated instrument loops. These techniques are commonly referred to as on-line monitoring (OLM) methods.

Traditionally, instruments must be recalibrated at each refueling outage in accordance with nuclear regulations. However, many utilities consider these mandatory periodic calibrations to be a suboptimal method. One concern with the periodic calibrations is that they only check the sensor's operating status at every fuel outage, meaning that faulty sensors may remain undetected for periods of up to 24 months. Also, studies have confirmed that it is very rare for a sensor to be found out of the required as-left as-found calibration range; typically less than 5% of the 50 to 150 calibrated instruments are found in degraded condition that require maintenance [EPRI 2004]. Additionally, OLM has the potential to bring about considerable direct and indirect cost savings by reducing the number of unnecessary calibrations and from the consequential reduction in outage time and lower chance of an unnecessary plant trip. OLM may also provide as low as reasonably achievable (ALARA) benefits, with more prompt discovery of instrument problems and opportunities for more timely and convenient corrective action. Furthermore, one of the greatest concerns with the periodic calibrations is that performing maintenance on components that are operating correctly provides an opportunity for a fault to enter the system. Instead, the OLM methods will monitor the instrument channel condition and identify those that have degraded to an extent that warrants calibration. The identified instrument channels will be investigated and then possibly classified as either normal, needing calibration at the next outage, or as entirely inoperable; based on the degree of degradation. Proponents of these technologies expect on-line methods to reduce maintenance costs, reduce radiation exposure, reduce the potential for miscalibration, increase instrument reliability, and may, as a consequence, reduce equipment downtime.

Because of the clear advantages brought about by OLM, several nuclear utilities are involved with testing and implementing these methods. In England, Sizewell B has implemented OLM for sensor calibration extension which reduced their most recent refueling outage by several days. However, for U.S. utilities to receive the full benefits of OLM, they must apply for plant-specific license amendments that grant regulatory approval to relax the calibration frequency. An adequate review of the license amendments requires a sufficient background in the technical details of the OLM methods. In particular, a thorough understanding of the uncertainty analysis associated with the particular modeling techniques is required. This report was developed with the objective of providing the information needed to understand the critical technical issues associated with the current OLM methods.

Research and development during the 1990s resulted in Topical Report TR-104965, *On-Line Monitoring of Instrument Channel Performance*, developed by the Electric Power and Research Institute (EPRI) Utility On-Line Monitoring Working Group. This EPRI report focused on the generic application of on-line monitoring techniques to be used as a tool for assessing instrument performance. The report proposed a requirement to have one channel for each measured variable fully calibrated every refueling outage, with every channel fully calibrated every 8 years or less. This would result in calibration every 6 years for a three-channel measurement with a 24-month refueling cycle, and every 4 years for a two-channel measurement in a similar plant. The maximum interval would be 6 years for a four-channel variable in a plant with an 18-month refueling interval. In July 2000, the U.S. Nuclear Regulatory Commission (NRC) issued a safety evaluation report (SER), released in September 2000. The SER concluded that the generic concept of OLM for tracking instrument performance as discussed in the topical report was acceptable. However, they also listed 14 requirements that must be addressed by plant-specific license amendments if the TS-required calibration frequency of safety-related instrumentation is to be relaxed. A later version of the EPRI topical report addresses these 14 requirements [EPRI 2000]. In turn, this volume of the NUREG/CR presents an in-depth theoretical study and independent review of the most widely used OLM sensor calibration techniques. It includes a presentation of the theory and an evaluation of the assumptions inherent in the methods. It should not be construed as an NRC endorsement of any of the described methods or technologies.

The instrument channel calibration monitoring methods presented in this report are empirical in nature. Empirical models are used over first-principle models because first principle models have not yet been proven to provide the accuracy necessary for calibration verification. With empirical modeling, historical plant data are used to construct the predictive models. Plant data sampled from past operating conditions embody the normal relationships between the process variables. After the predictive models are constructed, they are put into a monitoring mode to provide the best estimates of process variables for previously unseen data.

Figure 1-1 is a simple block diagram of a basic instrument calibration monitoring system. In this figure, a vector of sensor measurements ($\boldsymbol{x}$) is input to a prediction model that calculates the best estimates of the sensors ($\boldsymbol{x}'$). The estimates are compared to the measured values, forming differences called residuals ($\boldsymbol{r}$). A decision logic module determines if the residuals are statistically different from zero and establishes the health or status ($\mathbf{s}$) of each sensor. This module may also use predictive uncertainty values and drift limits to determine the condition of the instrument channel. The first volume in this series addresses how to establish the maximum acceptable value of deviation (MAVD) and allowable deviation value for on-line monitoring (ADVOLM), which are the conservative limits that are used to identify the onset of a drift problem.

Originally, EPRI assumed that OLM would be performed continuously and wrote the EPRI-TR-10495 report to reflect this fact; however, several members of the EPRI user's group have shifted to a periodic procedure [EPRI 2000]. The SER stated that quarterly monitoring is acceptable, so it is probable that some plants applying for a license amendment will wish to perform OLM in off-line batch mode on a quarterly basis. However, this NUREG/CR advocates the use of continuous or near-continuous monitoring. Continuous or near-continuous monitoring offers an added level of safety because instrument channels are surveillanced more frequently and problems can be identified much sooner. With continuous monitoring, communications and capacity issues may arise, but can be combated with a superior computer and network infrastructure. Additionally, when not applying OLM continuously, an additional margin must be subtracted from the drift limits (ADVOLM and MAVD) equal to the amount an instrument may drift in the next 3 months. Care must be taken to make sure plants have accounted for this margin and that they understand that they cannot switch from a continuous to quarterly surveillance OLM system without recalculating the ADVOLM and MAVD.

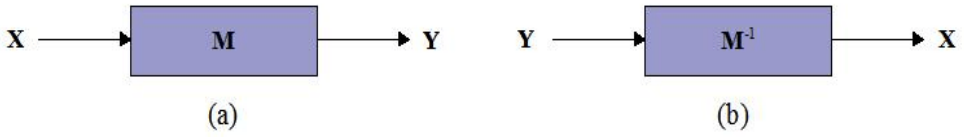


Fig. 1-1. Instrument calibration monitoring system diagram.

Although Fig. 1-1 may imply that OLM is a relatively simple process, it can actually be very complicated. There are many choices regarding the type of prediction model to use, and many of these models are quite complex. This report addresses three of the most widely employed nonredundant prediction models for OLM, namely, auto-associative neural networks (AANN), auto-associative kernel regression (AAKR), and auto-associative multivariate state estimation technique (AAMSET). All of these methods are plant-wide methods, meaning that they calculate the parameter estimate using the plant historical data from a group of correlated, but not necessarily redundant, instrument channels. These instruments may take measurements from any part of the plant's operation, although they are typically grouped by a single system or subsystem.

In contrast to the plant-wide model, redundant sensor models have been used to perform OLM. Redundant modeling techniques use only the measurements from a group of redundant instrument channels to obtain the parameter estimate. In this context, the term "redundant" describes instrument channels that measure the same process parameter over a similar operating range. An example of a redundant method would be simple averaging. Although redundant techniques are generally more intuitive and easier to troubleshoot than nonredundant techniques, they are unable to detect common mode failure and also may be more affected by spillover, which occurs when a faulty sensor input degrades the predictions for the other sensors in the group. Because of the disadvantages associated with the redundant models, the nonredundant models are becoming the dominating technique for OLM applications and are the focus of this report. However, the Instrumentation and Calibration Monitoring Program (ICMP), a redundant algorithm originally employed by several members of the EPRI Utility On-Line Monitoring Working Group, is briefly described in Chap. 2.

The three nonredundant methods highlighted in this report are by no means the only technologies that are viable for OLM applications. Rather, they are included simply because they are the techniques being currently employed by the utilities closest to filing for the first license amendment. The U.S. nuclear plants that have considered or are actively using OLM for equipment and instrumentation assessment (without the relaxed calibration frequency) include Limerick, Salem, Sequoyah, TMI, and VC Summer, all of which use a system produced by Expert Microsystems, Inc. (expmicrosys.com), and Harris and Palo Verde, which use a system developed by SmartSignal Inc. (smartsignal.com). The underlying algorithm for Expert Microsystems' software is the AAKR, while SmartSignal software utilizes AAMSET. An AANN OLM system was developed by researchers at the Halden Reactor Project (HRP) in Norway and has been tested by several nuclear facilities throughout Europe.

Again, many other algorithms are suitable for OLM, including several redundant techniques. Nuclear plants are in no way limited to the three methods described in this report. However, before filing for a license amendment to extend the calibration frequency with OLM, it is necessary for the details of the algorithm, especially the uncertainty analysis portion, to be fully understood and made available for regulatory review. The algorithms cannot be treated as a black box. The technical aspects must be known, so that safeguards can be put into place to account for any inherent deficiencies or limitations that the algorithm may have.

1.2 Organization of this Report

The goal of this report is to provide an independent evaluation of the most current OLM techniques that may eventually be used to reduce the required manual calibration frequency of safety critical instrument channels. The first section of the report gives a general overview of OLM. Next, in Chap. 2, the general properties of data-based models are described. After establishing the basic theory behind empirical modeling, equations are derived for the three modeling techniques that are most applicable to OLM systems: auto-associative neural networks (AANN), auto-associative kernel regression (AAKR), and auto-associative multivariate state estimation technique (AAMSET). The general framework for estimating a model's predictive uncertainty is then presented. Uncertainty is a key issue surrounding the regulatory approval of OLM. The discussion of uncertainty in Sect. 3.4 of EPRI TR-104965-R1 NRC SER argues that the past MSET performance and a Monte Carlo analysis will provide the required evidence [EPRI 2000]. However, determining the predictive uncertainty is much more complicated than was addressed in the EPRI paper. Thus, this report outlines the problem of estimating uncertainty in great detail and provides a set of well-defined equations and requirements that need to be addressed by OLM systems. The equations are derived for both the analytical and Monte Carlo method for estimating uncertainty. In Chap. 6, both simulated and actual nuclear plant data are used to compare the modeling techniques and the different uncertainty estimation methods. Finally, Chap. 7 concludes the report by listing some of the assumptions inherent in the OLM modeling process that could present potential problems if not handled properly. Many of these assumptions are concealed until an abnormality occurs in the OLM system, and they have not been fully explored. Thus, the third volume of this report will apply the modeling and uncertainty analysis techniques to a wide variety of plant data sets to investigate the effects of these modeling assumptions and limitations.

2. MODELING TECHNIQUES

2.1 Introduction

Modeling involves the use of a chosen technique to describe an event or situation. This introduction focuses on modeling a process by learning the relationships between process parameters and how this methodology relates to empirical modeling. There are two major system modeling categories: first-principles models and data-based (empirical) models. This section discusses the advantages and disadvantages of these two methods, explores data-based modeling in more detail, describes how data-based modeling relates to instrument calibration monitoring systems, reviews redundant sensor monitoring, and discusses the historical development of the three commercially available empirical modeling techniques.

2.1.1 First-principles vs data-based

Modern science and engineering is based on first-principle models (e.g., Newton's Laws). These models are usually developed during the design phase of developing a complex process plant. The major advantage of using first-principle models to describe process variable interactions is that they are based on physical equations. Usually, experimental data are used to verify the models, but there is a great deal of certainty in the accuracy of the models before a plant is ever built. First-principles models require substantial engineering time to develop; and in some instances, such as complex chemical reactions, accurate first-principles models are not available. Even if a highly accurate first-principles model is developed for a newly constructed process, operations will degrade or otherwise change the system from its design basis. In fact, first-principles models rarely incorporate all of the functionality of the system being modeled and do not provide the sensitivity necessary to detect slight changes in plant degradation.

Data-based models are constructed by collecting process parameter measurements recorded over the operating range of a plant. Shutdown and start-up data are not always included, but in general, monitoring over the entire operating range is beneficial because it will help reduce the single-point monitoring penalties which were discussed in Volume 1 of this series. The relationships between the sensor measurements are somehow learned or embedded in the model architecture. Because these relationships are usually nonlinear; flexible, nonlinear models are commonly used. These nonlinear, data-based systems are only accurate when applied to the same, or similar, operating conditions under which the data were collected. When plant conditions or operations change significantly, the model is forced to extrapolate outside the learned space, and the results cannot be trusted. Unlike linear models, which extrapolate in a known linear fashion, nonlinear models have poor extrapolation capabilities because it is impossible to predict how the system will respond outside of the previously observed operating conditions. Nonlinear models should be monitored for these conditions and should not be used outside their area of expertise. Systems designed to perform this function are commonly termed reliability assessment modules because they assess the reliability of the OLM system. Additionally, data-acquisition systems may not have acquired data on the conditions of interest, or the collected data may not have the accuracy and resolution required for the modeling task. Automated techniques and expert analysis based on human observation are needed to ensure that the collected data are suitable for OLM model construction and to ensure the usability of the modeling results.

The combination of data-based and first-principles models are termed hybrid models. In these designs, first-principles models are developed and improved with data-based models as data become available. These methods provide the robustness of first-principles models with the sensitivity of data-based models. This hybrid framework, although more complicated, has a very important advantage. Purely data-based systems are not reliable when the monitored process moves into new operating

conditions that may result from configuration changes, new operating practices, or external factors such as unusually cold cooling-water temperatures in condensers. Through proper application of hybrid systems, the predictions can be forced to revert toward the first-principles model when new operating conditions are encountered and will use the data-based models when in familiar operating conditions.

Several useful overview papers on hybrid modeling include Thompson and Kramer (1994), Wilson and Zorsetto (1997), and te Braake and van Can (1998). The hybrid approach has commonly been termed a "grey-box" modeling approach, as contrasted to "black box" modeling of neural networks or other data-based approaches. The term "grey-box" comes from the idea that a portion of the internal model (i.e., the first-principles model) is explainable. In other technical papers these approaches are termed "semi-mechanistic" models. Several of the papers on "semi-mechanistic" models focus on the development of accurate, robust, repeatable, empirical models, but all of the "semi-mechanistic" techniques can be applied to the data-based portion of hybrid modeling. Because hybrid modeling also requires the development of highly accurate first-principles models, it is not currently being tested in the nuclear industry and is beyond the scope of this report.

2.1.2 Learning from data

Modern complex systems almost always have an array of digital monitoring systems including transducers, signal processors, and displays. Many industries already employ complex modeling systems and have these digital monitoring systems. Some of the processors and displays found in these systems include either theoretical (first-principles) models or data-based models of the processes. For example, in addition to the normal (operational) indications of miles traveled, coolant temperature, oil pressure, fuel remaining, and electrical generator output, some automobiles include a simple data processing system that displays instantaneous and average miles per gallon, average speed, and estimated miles remaining until the gasoline tank is empty. Such a system is an example of the use of a first-principles model (involving simple mathematical operations) in which the miles traveled is divided by the change in fuel over a very short period of time or the lapsed time since the car started to produce the instantaneous and average miles per gallon. For such a model, the inputs would be the change in the amount of fuel and the elapsed time, while the output would be an estimate of the miles per gallon. Similarly, multiplication of the fuel remaining by the miles per gallon gives the estimated miles remaining before the tank is empty.

The above system is a classical example of a "direct" model in which measured inputs and a model are used to produce an output. Such a system is shown in Fig. 2-1(a), where $\boldsymbol{X}$ is the input, $\boldsymbol{Y}$ is the output, and $\boldsymbol{M}$ is the system matrix representing the system model. The input-output relationship for such a model is given by:

$$\boldsymbol{Y} = \boldsymbol{M}\,\boldsymbol{X}\,. \qquad (2.1.2.1)$$

If one knows the input $\boldsymbol{X}$ and the model matrix $\boldsymbol{M}$, one can solve for the output $\boldsymbol{Y}$. For instance, if the input is an impulse, and the matrix represents a stable second-order polynomial system, the output will be a smoothed sinusoidal function. The system matrix $\boldsymbol{M}$ is effectively a smoothing function that smoothes out the sinusoidal response to the input impulse [Schimek 2000].

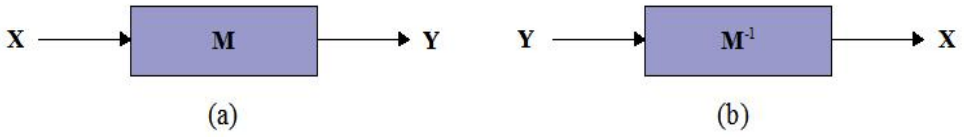


Fig. 2-1. Modeling diagrams for (a) a direct input-output system and (b) an inverse input-output system.

However, if one knows the output *Y* and the model matrix, ***M***, one must invert the above relationship to obtain the input ***X***, as follows:

$$\boldsymbol{X} = \boldsymbol{M}^{-1}\boldsymbol{Y} \quad . \tag{2.1.2.2}$$

This is generally known as an "inverse problem" because the normal input-output (i.e., the cause-and-effect) relationship, is reversed as shown in Fig. 2-1(b). In this case, the smoothed sinusoid is now the input, and the impulse is the output. Mathematically, this is possible if the system matrix M can be inverted. However, the inverted system matrix $\boldsymbol{M}^{-1}$ is now the inverse of a smoothing function, or a "roughening" function, in which small perturbations would be enlarged, producing an erratic influence on the prediction of ***X***. The presence of even a very small amount of noise in the input signal (***Y***) could result in an unstable situation. This is a result of the problem being ill-posed, a special type of problems discussed in detail in the next section. In most cases, the system matrix ***M*** probably cannot be inverted mathematically. As a result, the relationship implied by Eq. (2.1.2.2) would not be feasible. Even if it could be inverted, the presence of noise could produce meaningless results. Approximate methods of inverting a matrix exist and can sometimes be used to obtain reasonable results. However, as discussed below, care must be taken because the inverse problem may also be an "ill-posed" problem. Indeed most inverse problems (but not all) are ill-posed. Methods of dealing with this situation will be discussed extensively in the following section.

A similar inverse situation exists if one tries to obtain the system matrix ***M*** from the input ***X*** and the output ***Y***. The relationship now becomes:

$$\boldsymbol{M} = \boldsymbol{Y}\,\boldsymbol{X}^{-1} \quad . \tag{2.1.2.3}$$

If the input function *X* is unknown or cannot be inverted, the relationship of Eq. (2.1.2.3) is not valid and hence cannot be used. Furthermore, if either the input ***X*** or the output ***Y*** contains significant noise, the relationship of Eq. (2.1.2.3) could be seriously compromised. However, in some situations, where the input is known to be a good approximation to a white noise over the frequency range of interest, the relationship of Eq. (2.1.2.3) can be used because $\boldsymbol{X}^{-1}$ is constant over the range of interest. Hence ***M*** is proportional to ***Y***.

Most practical measurements in real systems have one feature in common: they involve, in some form, the process of inversion. The reason is that most measured quantities are outputs of the system, and the desired results are either the inputs or, more commonly, the system characteristics described by ***M***. This means going from effect to cause, for example, from a finite set of noisy (output) data to a general law governing this set of data or from a system's output to its input. In general terms, one has to reconstruct, from some manifestation of a state of nature or a system, the characteristics of this state of nature or system. Inverse problems per se have been known to mathematicians for many decades, but

only recently have methods developed in inverse theory been applied by engineers to real problems of surveillance, sensor validation, model building, and verification.

2.1.3 Ill-posed problems and regularization

For centuries, mathematicians have known that many very important practical problems are underdetermined. In other words, data do not provide enough information to determine a single unique solution or relationship. Examples of underdetermined problems are almost endless in every field, such as spectrum analysis, image reconstruction, interpolation and extrapolation from numerical data, factor analysis, solution of the diffusion equation backward in time, de-convolution, and neural network training. Such underdetermined problems have a special name: they are "ill-posed" problems or incorrectly posed problems in the sense that it is incorrect to ask somebody to solve a problem that has no unique solution. An important subclass of ill-posed problems is inverse problems that deal with the inversion of cause-effect relationships. Most inverse problems are ill-posed, and the terms "ill-posed" and "inverse" are often used as synonyms. However, it is important to understand that an inverse problem might not be an ill-posed problem, just as an ill-posed problem need not necessarily be an inverse problem.

Because the goal of empirical modeling is to learn relationships from historical data, it is by definition an inverse problem. Furthermore, since modeling techniques will never have a "complete" set of information and will never be perfectly specified (i.e., an empirical model's input-output relationship will never be exactly correct), ill-posed problems are ubiquitous. How does this affect instrument calibration validation systems? Because uncertainty is fundamentally related to a model's stability (i.e., low uncertainty implies a stable model), the identification and management of ill-posed problems is extremely important.

In the fields of biology, econometrics, geophysics, machine and plant diagnostics, and surveillance, truly well-posed problems are practically unknown because the true input parameter values are not known; only their measurements are known. In spite of the prevalence of such ill-posed problems, the implications and significance of inverse and ill-posed problems are often not fully understood and appreciated by the engineering community. Until recently, the vast majority of practicing engineers was unaware of inverse problems and hence did not have the type of "inverse" mentality necessary in resolving practical engineering problems.

During the past decade much research has been performed on methods that deal with ill-posed and inverse problems in engineering. It has become clear that the ability to reason and build models in under-determined situations is critical; even more critical is the ability to decide which of several competing models is best. Unfortunately, raw data are of limited use in such situations because they may point to a number of models or to a unique model that is incorrect.

Much of the work done in inverse theory can be considered to be a special case of inductive inference, which is the process of inferring a general law, or model, from a finite set of observable instances. An attempt to apply methods of inverse theory to the process of model development is a natural next step. The greatest contribution of inverse theory to the solution of the problem of inductive inference has been the introduction of "regularization" methods. Regularization is the procedure used to augment information in the data with some additional information about the sought solution that might be available from outside the collected data. A classic example of regularization is the solution of Fredholm integral equations, where information contained in the data is augmented by an additional assumption about the "smoothness" of a sought solution. Another example, from statistics, is ridge regression [Hoerl 1970] where instead of minimizing a "self-evident" least-squares function, one minimizes a "nonself-evident" cost function consisting of a combination (sum) of two terms: a least-squares term and norm or semi-norm of the vector of regression coefficients. This idea of augmenting the data by some additional information has been implemented in a number of methods and under different names to solve

the problem of induction. Different methods to produce consistent, reliable, and low-noise results, such as numerical inversion like Tikhonov regularization [Tikhonov 1963], truncated single value decomposition (TSVD), and ridge regression, as well as such methods' applicability to experimental measurements, will now be discussed.

Hadamard [1923] defined a well-posed problem as a problem that satisfies the three following conditions:

1. A solution for the problem exists.
2. The solution is unique.
3. The solution is stable or smooth under small perturbations of the data (i.e., small perturbations in the data should produce only small perturbations in the solution).

If any of these conditions are not met, the problem is termed "ill-posed," and special considerations must be taken to ensure consistent reliable solutions. To understand the essence of ill-posed problems, consider the linear least-squares model, whose objective is to find a linear combination of predictor variables **X** that accurately models the response variable **Y**.

The ordinary least-squares (OLS) solution is given by:

$$\boldsymbol{w} = \left(\boldsymbol{X}^T \boldsymbol{X} \right)^{-1} \boldsymbol{X}^T \boldsymbol{Y} \ , \tag{2.1.3.1}$$

which minimizes the ordinary least-squares function:

$$\boldsymbol{E}(\boldsymbol{w}) = \left(\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{w} \right)^T \left(\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{w} \right) \ . \tag{2.1.3.2}$$

This solution involves the inversion of what is commonly called the Fisher information matrix, $\boldsymbol{X}^{\mathrm{T}}\,\boldsymbol{X}$. If the predictors ($\boldsymbol{X}$) are highly correlated, this matrix is ill-conditioned, and small changes in the predictor data cause significant changes in the solution weights ($\boldsymbol{w}$).

A measure of condition is the ratio of the largest Eigen value to the smallest Eigen value; this ratio is commonly called the "condition number." The numerical inversion of ill-conditioned matrices causes unstable solutions, which means that a problem is ill-posed. The difficulty of constructing prediction models with correlated data is not restricted to linear regression models but also occurs, with even greater instabilities, in such nonlinear modeling techniques as neural networks.

2.1.4 Parametric vs nonparametric models

An empirical model's architecture may be either defined by a set of parameters and functional relationships (parametric) or a set of data and algorithmic estimation procedures (nonparametric). In a parametric model, training data are used to fit the model to the data according to a predefined mathematical structure. For example, consider the following polynomial model:

$$\boldsymbol{y} = \boldsymbol{b}_0 + \boldsymbol{b}_1\,\boldsymbol{x}_1 + \boldsymbol{b}_2\,\boldsymbol{x}_2 + \boldsymbol{b}_3\,\boldsymbol{x}_1\,\boldsymbol{x}_2 + \boldsymbol{b}_4\,\boldsymbol{x}_1^2 + \boldsymbol{b}_5\,\boldsymbol{x}_2^2 \ . \tag{2.1.4.1}$$

To completely define this model for a given set of training observations, the polynomial coefficients bi are varied until they minimize some objective function, usually the sum of the squared error (SSE). Once the optimal polynomial coefficients have been estimated, the model is completely specified by

Eq. (2.1.4.1) and the estimated coefficients. Therefore, a parametric model may be roughly defined as a model that can be completely specified by a set of parameters and a functional relationship for applying these parameters to new data in order to estimate the response.

A nonparametric model, by contrast, stores past data exemplars in memory and processes them when a new query is made. For instance, rather than modeling a whole input space with a parametric model such as a neural network or linear regression, local nonparametric techniques may be used to construct a local model in the immediate region of the query. These models are constructed "on the fly" not beforehand. When the query is made, the algorithm locates training exemplars in its vicinity and performs a weighted regression with the nearby observations. The observations are weighted with respect to their proximity to the query point. To construct a robust local model, one must define a distance function to measure what is considered to be local to the query, implement locally weighted regression, and in some cases consider additional regularization techniques.

Several trade-offs need to be considered before deciding whether a parametric or nonparametric model should be used. One of the most obvious features of a parametric model is that its architecture and behavior are well defined. This feature is both an advantage and disadvantage. On one hand, by knowing the parameters and functional relationship, the parametric architecture may be very small, compared to the training data. Additionally, the model is easily conceptualized because most engineers and scientists have been exposed to many of the functions that may be used, while nonparametric techniques require a level of abstraction because the model will be different for different query vectors. Finally, parametric models tend to require less time to estimate an output value, since its structure has been defined up-front, whereas in a nonparametric model, a new structure must be inferred for each query vector.

On the other hand, when a model is restricted to a specific functional relationship, if the data do not follow this relationship, its estimates are inherently biased. In either modeling case, if the system being modeled moves into a new operating region, the functional relationship may no longer hold, or the model may need to be retrained. The issue of retraining is not necessarily problematic in most parametric techniques, but it can lead to problems when complex nonlinear techniques such as neural networks are used, because training may require extensive computational effort and may produce inconsistent results as a result of the randomization used in network initialization and inherent local minima. Conversely, a nonparametric model may continuously adapt to additional operating regions because new exemplary observations may be easily added to the training data, which is often used to create the model's "memory" matrix, usually a subset of the training data. However, in OLM applications, the model developer should limit retraining to times when the situation is fully investigated and it is clear why the retraining is necessary. This investigation is necessary to ensure that the model does not learn drifts or other anomalies as if they were normal operations.

2.1.5 Inferential, heteroassociative, and auto-associative models

An additional classification used to describe an empirical model, is according to whether it is inferential, heteroassociative, or auto-associative. An inferential model uses a set of predictor variables to infer the values of a response variable [see Fig. 2-2(a)]. For example, the temperature change of the water for each channel in a nuclear reactor fuel assembly may be used to infer the heat generated by the entire assembly. This model may be expanded to a heteroassociative architecture having multiple inputs and outputs, where the response variables are the heat generated for each channel [see Fig. 2-2(b)]. The distinction between inferential and heteroassociative models is that inferential models produce a single estimate, while a heterassociative model is able to produce multiple parameter estimates. Inferential and heteroassociative models are generally applied to situations where a dependency is known and a parameter's measured value is compared to a model's predictions to identify a change in the dependency (an example is inferential estimation of nuclear power plant feedwater flowrate in order to monitor venturi fouling [Gribok et al. 2004]).

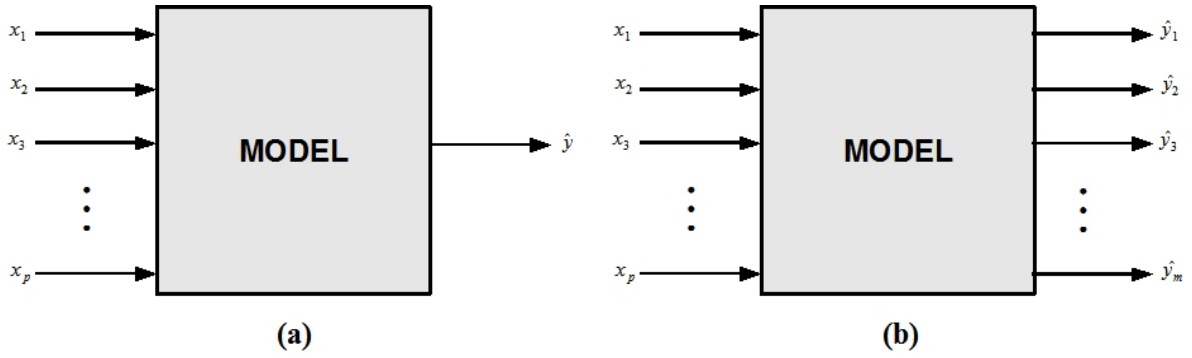


Fig. 2-2. Schematic diagram of (a) inferential and (b) heteroassociative models.

An auto-associative model is a model whose outputs are trained to emulate its inputs over an appropriate dynamic range (see Fig, 2-3). Since measured plant data are often corrupted by noise, faults, and/or biases; an auto-associative model may be used to estimate the true parameter values. Here, "true" refers to the values collected during normal operations used to train the model, or denoised versions of those fault-free measurements. In other words, an auto-associative model will estimate the correct input values using the correlations (relationships) embedded in the model during its training. The estimated correct value from the auto-associative model can then be compared to the actual process parameter to determine if a sensor has drifted or has been degraded by another fault type.

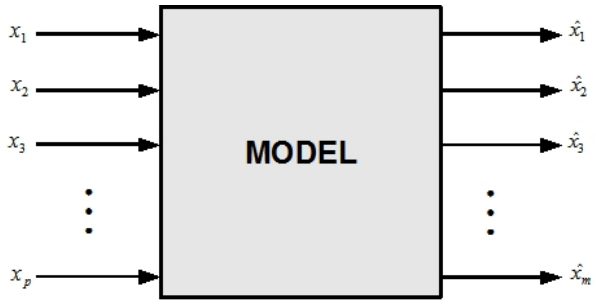


Fig. 2-3. Schematic diagram for auto-associative model.

2.1.6 Empirical modeling in calibration monitoring systems

Now that the generalities of empirical modeling have been discussed, how does empirical modeling fit into a calibration monitoring system? This relationship is explained by first considering an instrument calibration monitoring system's process diagram, as presented in Fig. 2-4.

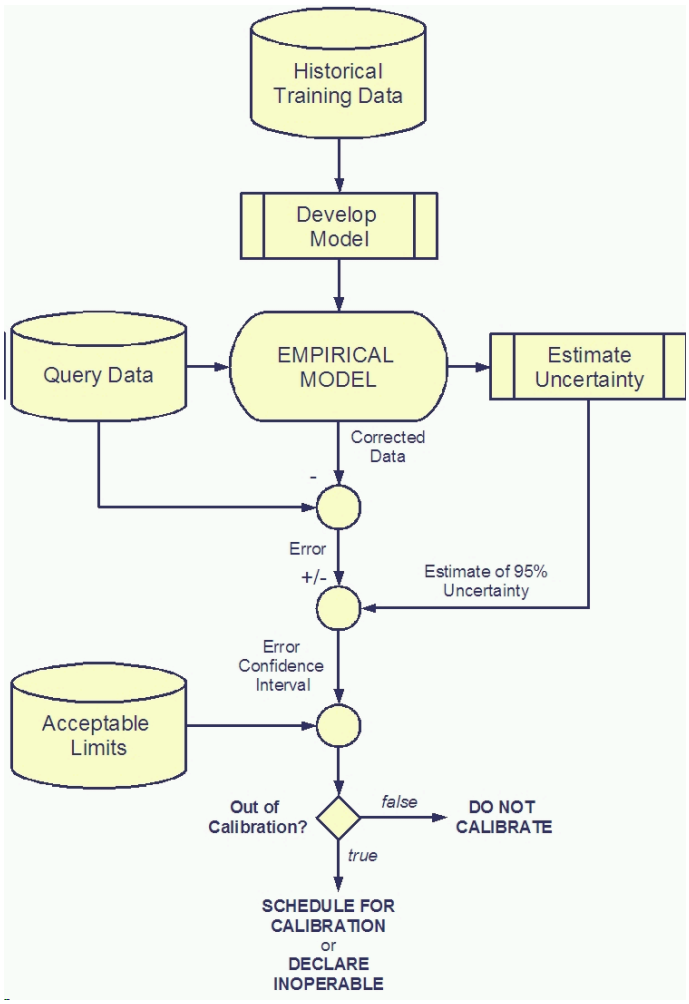


Fig. 2-4. Process diagram for generic calibration monitoring system.

It can be seen that the first step in developing a calibration monitoring system is to collect training data and develop an empirical model. Next, process measurements, typically termed query data, are presented to the developed model, which corrects the data. The measured values are subtracted from the corrected values (where corrected values refer to a model's estimates), providing the model's prediction error or residual. In addition, the 95% uncertainty, or variance, of the model's estimates is calculated and combined with the error to form its 95% confidence interval (CI). This CI is then compared to acceptable calibration limits, and if the limit is exceeded, then the instrument channel's condition must be investigated. After this investigation, one of three possible outcomes will result:

1. do not calibrate,
2. schedule a calibration, or
3. declare the sensor to be inoperable.

The level settings, the investigation details, and the procedure for making calibration decisions are expected to be plant specific and defined in a plant's license amendments. Specifically, Technical Specifications will need to be modified to accommodate the altered test intervals. However, EPRI's OLM user group recommends only implementing sensor calibration extension when the modeling error is smaller than the drift uncertainties so that set point calculations will remain unmodified. Other utilities

may choose to recalculate set points by incorporating the prediction uncertainties into the set point calculations, but this has not bee investigated to date.

From this discussion, it is clear that the empirical model performs the most critical task of the calibration monitoring system by computing the process parameter estimate for a given system state. In addition, notice that the auto-associative architecture is well-suited for OLM because of the "correcting action" it performs. With this correcting action, the model estimates the "error-free" value, and this estimate can be used to calculate the residual that is used with the uncertainty, sensitivity measurements, and drift limits to determine the calibration status of each sensor.

2.1.7 Review of redundant sensor monitoring—ICMP

The Instrumentation and Calibration Monitoring Program (ICMP) algorithm is used for OLM of redundant sensor groups. It was developed by EPRI in the early 1990s and essentially was the original method used to perform OLM for sensor calibration monitoring [EPRI 1993]. In ICMP, a weighted averaging algorithm is used to determine an estimate of the true process parameter. The ICMP algorithm assigns a consistency value, $\boldsymbol{C}_i$, to each of the signals for each data sample evaluated. This consistency value denotes how much of the signal's measured value contributes to the process estimate. The value is based on the absolute difference between a given sensor and other sensors in the group. Thus, inconsistent signals contribute less to the process estimate. For example, for a group of three redundant sensors, the constituency value compares the output of each instrument to the output of the other two instruments. If the *i*th instrument's output is sufficiently close to the output of both of the other instruments, its consistency value, $\boldsymbol{C}_i$, will be 2. However, if the *i*th instrument's output is only sufficiently close to one of the other instruments, then the $\boldsymbol{C}_i$ will be 1. If the *i*th instrument's output is not close to either of the two remaining instruments, then the consistency value, $\boldsymbol{C}_i$, will be 0. Overall, if a signal agrees within a tolerance to another signal in the group it is declared to be consistent, and the consistency value for that signal, $\boldsymbol{C}_i$, is found with Eq. (2.1.7.1).

$$\text{If } \left|\boldsymbol{m}_i - \boldsymbol{m}_j\right| < \delta_i + \delta_j \text{ , then } \boldsymbol{C}_i = \boldsymbol{C}_i + 1 \quad , \tag{2.1.7.1}$$

where

$\boldsymbol{C}_i$ = the consistency value of the *i*th signal,
$\boldsymbol{m}_i$ = signal ***i***'s measured value,
$\boldsymbol{m}_j$ = signal ***j***'s measured value,
δ_i = the consistency check allowance for instrument ***i***,
δ_j = the consistency check allowance for instrument ***j***.

The values for the consistency check allowances are dependent on the uncertainty present in the signals, such as

$$\left|\boldsymbol{m}_i - \boldsymbol{m}_j\right| \le 2\delta \ . \tag{2.1.7.2}$$

After the consistency values are calculated, the ICMP parameter estimate can be calculated as

$$\hat{\boldsymbol{x}} = \frac{\sum_{i=1}^{n} \boldsymbol{w}_i \boldsymbol{C}_i \boldsymbol{m}_i}{\sum_{i=1}^{n} \boldsymbol{w}_i \boldsymbol{C}_i} \ , \tag{2.1.7.3}$$

where

$\hat{x}$ = the ICMP parameter estimate for the given data sample,

w_i = the weight associated with the ith signal.

The weight values are included to allow the user to apply a greater weighting to more accurate or reliable sensors within a redundant group. If there is no preference within the group, all weight values can be set to 1, reducing the equation to

$$\hat{x} = \frac{\sum_{i=1}^{n} C_i m_i}{\sum_{i=1}^{n} C_i} \quad . \tag{2.1.7.4}$$

The consistency check factor controls the influence of an individual signal on the ICMP parameter estimate. If all sensors are considered equally consistent, the ICMP estimate is just the simple average of the redundant sensors. If a sensor's consistency value is zero, it will not influence the parameter estimate. If all sensors are inconsistent, the parameter estimate is undefined.

Once the parameter estimate is calculated, the ICMP algorithm evaluates the performance of each individual sensor relative to the parameter estimate. This is done through the use of an acceptance criterion.

$$\text{If, } \left|\hat{x} - m_i\right| \geq \alpha_i \text{ , then } m_i \text{ has potentially drifted beyond desired limits ,} \tag{2.1.7.5}$$

where α_i = the acceptance criterion for the ith signal.

There may be a different acceptance criterion related to calibration or operability. When the deviation between a sensor's measurement and the current parameter estimate exceeds the acceptance criterion, that sensor is considered to have drifted out of calibration. At this point the sensor is assumed to have failed. Note that failing the acceptance criterion does not necessarily disallow the failed sensor's value to influence the ICMP estimate. The consistency check factor must also be exceeded, and it is not necessarily related to the acceptance criterion. The 2002 paper "Monte Carlo Analysis and Evaluation of the Instrumentation and Calibration Monitoring Program" further details the relationship between the acceptance criteria and the consistency check factor and also provides numerical examples of the ICMP algorithm for a variety of redundant sensor groups [Rasmussen 2002].

ICMP software was successfully installed at the Catawba and V.C. Summer Nuclear Stations [EPRI 2000]. Although these plants were using ICMP only as a performance monitoring and troubleshooting tool, they obtained positive results. These results helped to verify ICMP's diagnostic capabilities. The plants did note some of ICMP's inherent shortcomings. For instance, ICMP performed poorly when there was limited redundant instrumentation, as is normally found on the secondary side of a nuclear plant. ICMP is also unable to detect common mode drift failure (where all redundant instruments drift in the same direction at the same rate) during plant operation; whereas nonredundant models are able to detect this type of failure. However, as current calibration practices offer limited protection against common mode failure while the plant is operating, ICMP's inability to do so should not invalidate the technique. Since in OLM, at least one sensor will still be required to be calibrated during each outage, the same protection against common mode sensor failure that is provided by current calibration exists with ICMP. Typically, common mode failure of rack instrumentation would be detected every 3 months, and sensor common mode failure would be detected during manual calibrations, which are required to be performed

during each refueling outage. Nonredundant OLM techniques can detect common mode failure at the monitoring interval, which could be continuous. Still, due to some of ICMP's limitations, most plants (including V.C. Summer and Catawba) have migrated to the more advanced OLM modeling techniques, such as AANN, AAKR, and AAMSET. For more information about the algorithm and how to quantify its associated uncertainty see *Monte Carlo Simulation and Uncertainty Analysis of the Instrument Calibration and Monitoring Program* [EPRI 1995] and *Monte Carlo Analysis and Evaluation of the Instrumentation and Calibration Monitoring Program* [Rasmussen 2002].

2.1.8 Historical development

Early pioneers in the use of advanced information-processing techniques for instrument condition monitoring included researchers at the University of Tennessee–Knoxville (UT) and Argonne National Laboratory. Dr. Belle Upadhyaya was one of the original investigators in the early 1980s [Upadhyaya 1985, 1989, 1992] through a Department of Energy-funded research project to investigate the application of artificial intelligence techniques to nuclear power plants. Researchers at Argonne National Laboratory continued with similar research from the late 1980s [Mott 1987], when they developed the Multivariate State Estimation System (MSET), which has gained wide interest by U.S. nuclear utilities. SmartSignal Corporation of Lisle, Illinois, licensed the MSET technology for applications in all industries and subsequently extended and modified the basic MSET technology in developing its commercial Equipment Condition Monitoring (SmartSignal *e*CM™) software [Wegerich 2001]. The Electric Power Research Institute (EPRI) has used a product from Expert Microsystems called SureSense [Bickford 2003], which uses a similar kernel-based algorithm. The major European participant in this area is the Halden Research Project, in which Dr. Paolo Fantoni and his multinational research team have developed a system termed Plant Evaluation and Analysis by Neural Operators (PEANO) [Fantoni 1998, 1999]; it uses auto-associative neural networks (AANN) and applies them to the monitoring of nuclear power plant sensors.

Currently four major service providers and systems have used empirical modeling for sensor calibration monitoring in nuclear power plants:

1. Expert Microsystems has developed a system termed SureSense® Diagnostic Monitoring Studio® software, which currently uses a form of kernel regression called the Expert State Estimation Engine (ESEE).
2. SmartSignal Inc. has developed a product termed eCM™ using MSET for equipment condition monitoring.
3. The Halden Reactor Project (HRP) has developed a product termed Process Evaluation and Analysis by Neural Operators (PEANO) using AANNs.
4. Analysis and Measurement Services (AMS) Corporation has developed an averaging technique that is being used at Sizewell B in England for calibration extension. AMS has also recently been granted (2007) a DOE Phase II SBIR to develop and implement advanced and integrated techniques to monitor nuclear power plant instrumentation.

Expert Microsystems licensed an MSET-based monitoring system from Argonne National Laboratory and teamed with EPRI to develop and test a sensor calibration monitoring system for nuclear power plant implementation. Recently (2004), intellectual property issues over MSET have prompted Expert Microsystem to replace the MSET algorithm with another kernel technique termed Expert State Estimation Engine (ESEE) in their SureSense® Diagnostic Monitoring Studio® software. The software has a general framework, and other model architectures can also be implemented. The EPRI implementation program tested the SureSense® system on six nuclear power plants including Harris, Limerick, Salem, Sequoyah, TMI, and V.C. Summer.

SmartSignal also licensed the MSET technology from Argonne National Laboratory and purchased some of the seminal patents. They have implemented their system on all three Palo Verde Nuclear Units for equipment condition monitoring. The technology has detected equipment abnormalities such as an RCS hot leg temperature sensor failure and a reactor coolant pump-seal failure [Coppock 2003]. At the time of this writing, Arizona Public Service has not decided whether the cost-benefit analysis warrants a license amendment for calibration extension.

Researchers at the HRP in Norway have been leaders in the development of sensor calibration monitoring systems since their introduction of their system entitled Process Evaluation and Analysis by Neural Operators (PEANO) in 1995. This system was originally based on the AANN model [Fantoni 2002], but other models have also been implemented. The system utilizes a client/server architecture and a modular modeling structure. HRP researchers have applied the system to various plants around the world for equipment condition monitoring and sensor calibration monitoring. An excellent and complete history of PEANO's 15 years of development and implementation is provided by Fantoni [Fantoni 2005].

In the following sections, the fundamentals of the three commercially available empirical modeling techniques for sensor calibration monitoring will be discussed, specifically auto-associative neural networks (AANN), auto-associative kernel regression (AAKR), and auto-associative MSET (AAMSET). Although these products may be available commercially, this NUREG/CR tries to address the modeling techniques used in the products, and not the products themselves. The commercial product and the modeling technique are not the same thing and should not be confused. In fact, due to advances in technology or intellectual property disputes, several providers have been known to change the modeling technique they employee in their products. Thus, it is important to understand and separate the actual model used in the product, and not just associate each model with a single product.

2.2 Auto-Associative Neural Networks

To preface this section, it should be stated that at the time of this writing (2007), the NRC believes ANNs to be inherently nondeterministic and not sufficiently predictable for use in key safety functions such as sensor calibration monitoring. However, because they have historically been investigated for use in OLM systems, they will be included in this report.

2.2.1 Artificial neuron

Artificial neural networks (ANN) are highly complex, interconnected, nonlinear modeling techniques that have the flexibility to model any nonlinear relationship. ANNs are built from individual computing elements, called artificial neurons. Artificial neurons can be understood in terms of their biological counterparts. It can be seen in Fig. 2-5, a neuron's anatomy is composed of four major structures: the dendrites, the soma, the axon, and the synaptic terminals.

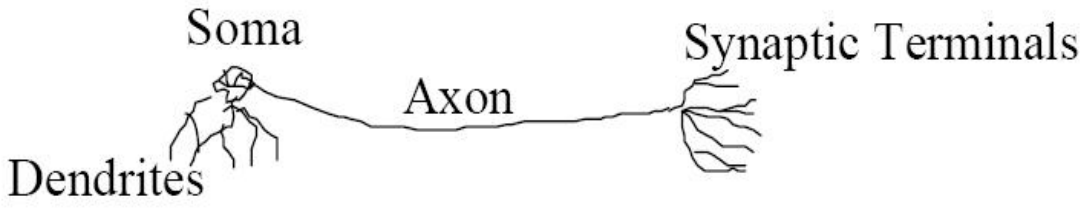


Fig. 2-5. Anatomy of biological neuron.

The body of the nerve cell is the soma. It receives and combines signals from other neurons through paths called dendrites (receptive zones). If the combined signal is strong enough it fires an output through an axon (transmission line). Synapses are areas of contact between neurons, they can be either excitatory or inhibitory; they regulate how much of each incoming signal passes to the neuron.

The anatomy of an artificial neuron is presented in Fig. 2-6. The inputs are signals that enter the neuron. Weights modify the incoming signals acting as the connection's synapse. The processing unit sums the modified inputs and modifies the result with an activation function. Finally, the output transmits the signal either to another neuron or out of the network.

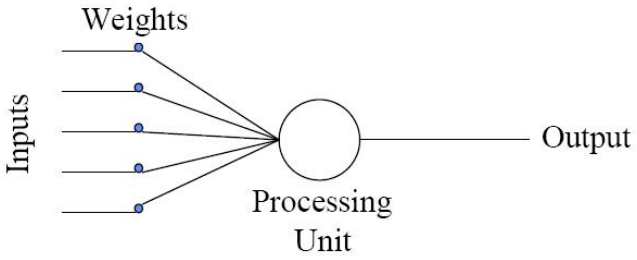


Fig. 2-6. Anatomy of an artificial neuron.

Recall that the processing unit of an artificial neuron performs two tasks: (1) it sums the weighted inputs and (2) modifies this result with an activation function. These tasks may be used to create a more detailed schematic for the processing unit, as seen in Fig. 2-7.

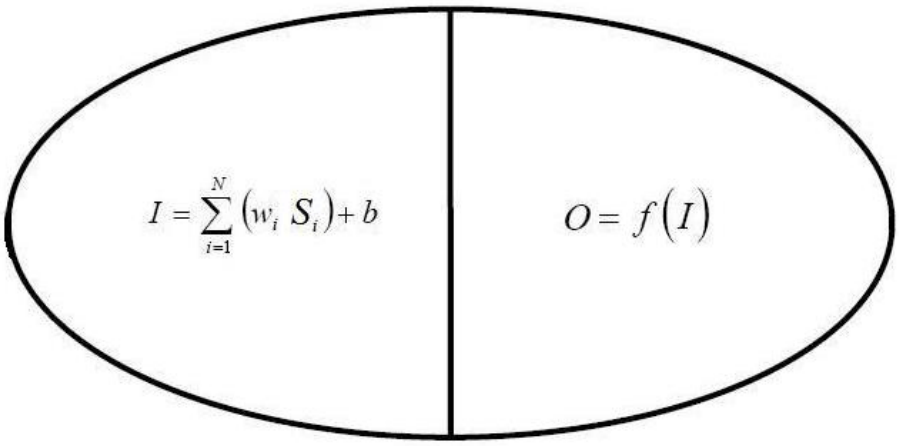


Fig. 2-7. Anatomy of the processing unit of an artificial neuron.

Here, ***Si*** is one of the ***N*** neuron inputs and $\boldsymbol{w_i}$ is its associated connection weight; ***b*** is the neuron's bias, ***I*** is called the neuron's internal activation, and ***O*** is the neuron output. Notice that the neuron output is a function of its internal activation, ***I***. This function is known as the neuron's activation function, and it may be either linear or nonlinear. Common activation functions include thresholds, signum, linear

(neuron output is equal to its internal activation), logistic, and hyperbolic tangent [Tsoukalas and Uhrig 1997].

2.2.2 Neural network architectures

A neural network may be loosely described as being a collection of many simple, but highly interconnected artificial neurons. Although this description is relatively simple, neural networks may have infinitely complex architectures. In general, there are three fundamental network architectures:

1. single-layer feed-forward networks,
2. multilayer feed-forward networks, and
3. recurrent networks

In a single-layer feed-forward network, neurons are collected into a single layer, and all connections feed forward through the network [see Fig. 2-8(a)]. In a multilayer feed-forward network, neurons are organized into multiple layers, and all connections feed forward through the network [see Fig. 2-8(b)]. Finally, recurrent networks are networks that have feedback connections (i.e., a neuron's input is affected by its previous output) and may exhibit temporal behavior [see Fig. 2-8(c)]. Each of these architectures has been prolifically adapted to solve many science and engineering problems.

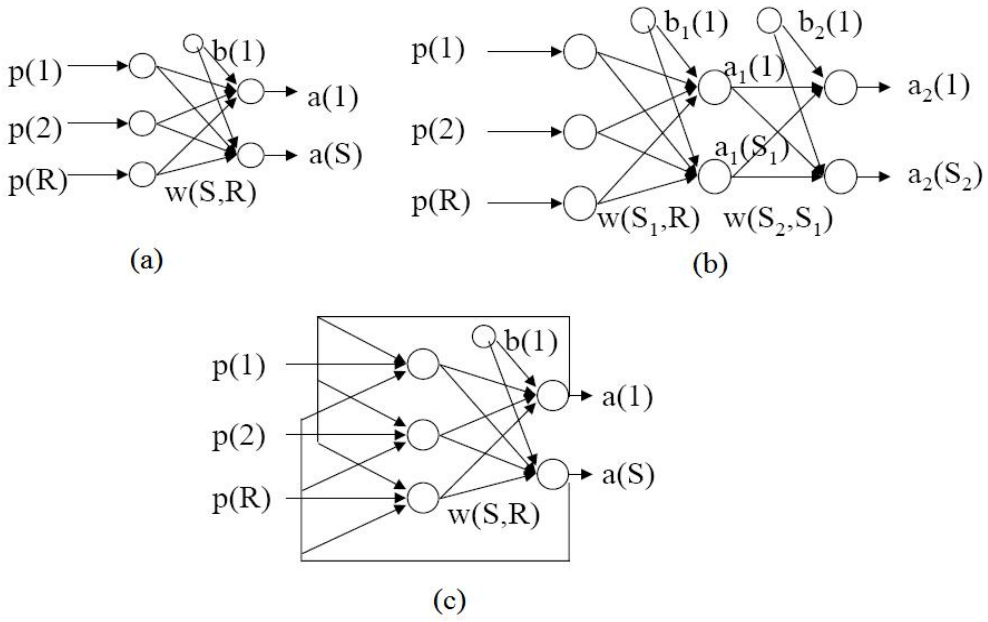


Fig. 2-8. Examples of (a) single-layer feed-forward, (b) multilayer feed-forward, and (c) recurrent network architectures.

2.2.3 Network training

Neural networks are generally trained in an iterative, batch manner. This process is illustrated in Fig. 2-9 and is summarized by the following steps.

1. Initialize neural network weights and biases to small random values and set iteration counter to zero.

2. Simulate network with training data.
3. Calculate the error of the network's estimates with their target values.
4. Compare error with a user-specified target value.
5. If the network does not meet the target performance criteria and if the maximum number of iterations has not been exceeded go to (a); otherwise go to (b).
   (a) Calculate updated weight and bias values, update the network, increment iteration counter, and go to step 2.
   (b) Stop training.

The different network training algorithms differ primarily in the way in which the weight and bias updates are determined. These training algorithms are all optimization techniques, and several algorithms exist that optimize the weights and biases to minimize the training error. Currently accepted methods for updating the neural network weight and bias values include gradient descent, conjugate gradient, one-step secant, Newton's method, and Levenberg-Marquardt [Bello et al. 1992; Demuth et al. 2005; Hagan et al. 1995; Snyman 2005]. Even though there are slight differences between the listed updating methods, the process is conceptually the same in that each combines a network's current state with a perturbed state (i.e., perturbed toward a minimum error) to develop the updated network parameters. The technique chosen is not important, only the resulting performance.

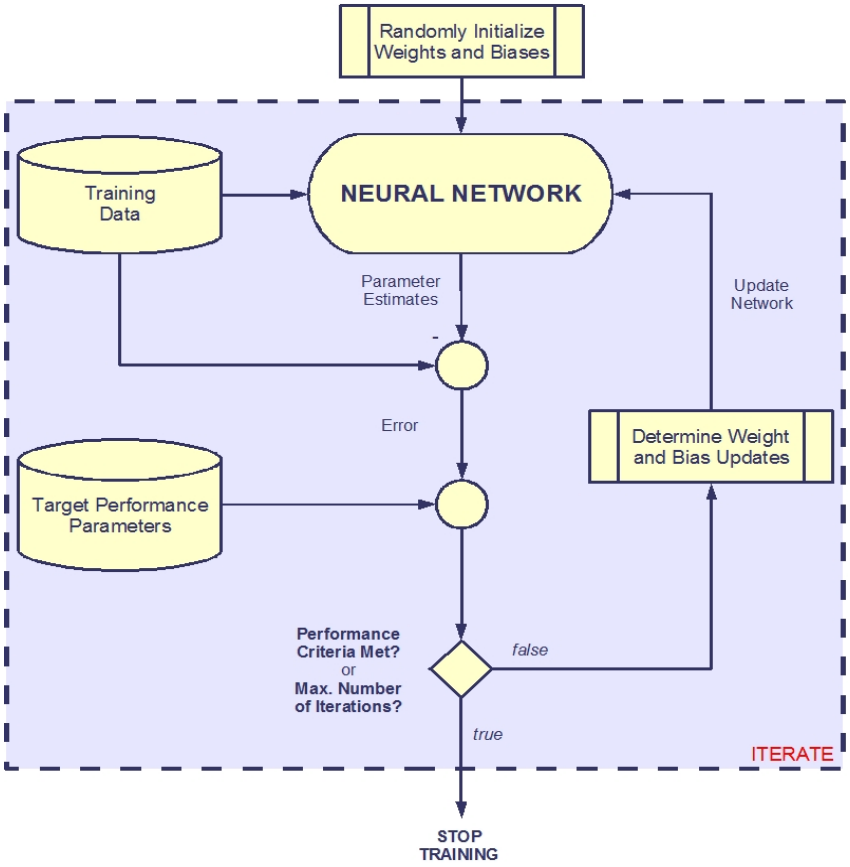


Fig. 2-9. Process diagram for general neural network training.

2.2.4 Historical development of the AANN

According to its original designer, Kramer [1991, 1992] states that the AANN was primarily developed to allow for the replacement of the conventional, multistep process of data rectification, gross error detection, failure identification, and sensor value replacement estimation commonly used in the monitoring of chemical processes with a single forward pass through the AANN. Prior to the development of the currently accepted AANN architecture, the concept of neural network autoassociation had been unsuccessfully investigated by Ackley et al. [1985], Rumelhart et al. [1986], Cottrell et al. [1987], and Chauvin [1989]. However, associative memory networks, which recall historical exemplars when supplied with new, similar patterns, were successfully developed by Hopfield [1982], Lippmann [1987], and Kosko [1988]. While the objectives of AANNs and associative memory networks are similar, AANNs "perform functional mappings while associative memory networks are classifiers, recalling a stored exemplar that most closely resembles a partial or corrupted input pattern" [Kramer 1992]. Although associative memory networks do not technically perform the task of autoassociation, they represent a step in the right direction. The breakthrough came when the "bottleneck" layer was introduced.

The bottleneck layer is defined as being a constraining layer having the smallest dimension of any layer in the entire network. Qualitatively, this layer acts as a nonlinear feature extraction or data compression agent, forcing data to be represented by fewer components than in its original state. For linear systems, this approach is well known and documented as principal component analysis (PCA) [Fukunaga and Koontz 1970], which is based upon a linear mapping between $\boldsymbol{p}$ and $\boldsymbol{A}$-dimensional spaces ($\boldsymbol{p} > \boldsymbol{A}$).

PCA is a method for explaining the variance of the input variable matrix in terms of a number of new latent variables called principal components (PCs) [Hodouin 1993]. PCA defines the closest plane to a system of points in space such that the sum of squares of the perpendicular residuals onto that plane was the least [Hotelling 1933]. In the $\boldsymbol{m}$-dimensional input variable space, the first loading vector $\boldsymbol{a_1}$ defines the direction of greatest variability, and the score vector $\boldsymbol{z_1}$ represents the projection of each observation vector (input variable pattern or sample) onto $\boldsymbol{a_1}$. Alternatively, $\boldsymbol{a_1}$ and $\boldsymbol{z_1}$ are the first eigenvectors of $\boldsymbol{X^T X}$ and $\boldsymbol{X X^T}$, respectively. The second PC is that linear combination of the input variables explaining the next greatest amount of variability subject to the condition that it is orthogonal to the first PC, that is $\boldsymbol{z_2} = \boldsymbol{E_1 a_2}$, where $\boldsymbol{E_1} = \boldsymbol{X} - \boldsymbol{z_1 a_1^T}$ is the residual matrix left after removing the predictions of the first PC. This process can be repeated until $\boldsymbol{p}$ PCs are obtained. Considering all of the PCs, the dimensionality of the input variables has not been reduced, but the axes of the input variable matrix have been rotated to a new orthogonal basis. With large data sets of correlated variables, after defining $\boldsymbol{A}$ PCs, where $\mathbf{A} << \boldsymbol{p}$, most of the variation in the input variable matrix ($\boldsymbol{X}$) can be explained by the first $\boldsymbol{A}$ PCs. This leads to a dimensionality reduction from $\boldsymbol{p}$ to $\boldsymbol{A}$. The similarity of the AANN architecture to PCA eventually led to the development of a nonlinear PCA (NLPCA) algorithm by Dong and McAvoy [1994, 1996].

2.2.5 AANN architecture

The general architecture of an AANN is presented in Fig. 2-10, along with the neuron symbol definitions in Fig. 2-11. It can be seen that there are four major neuron layers, specifically mapping, bottleneck, de-mapping, and output layers.

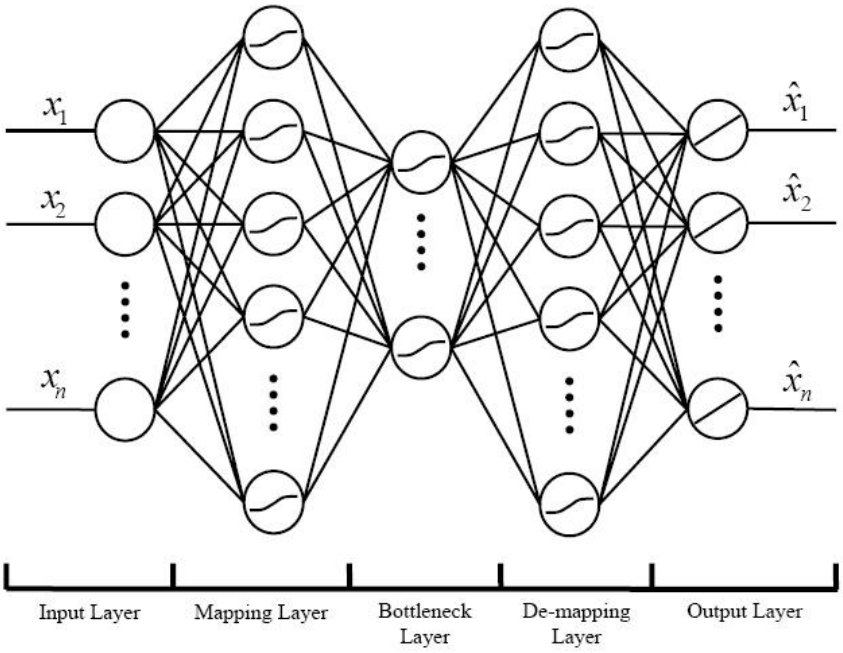


Fig. 2-10. AANN architecture.

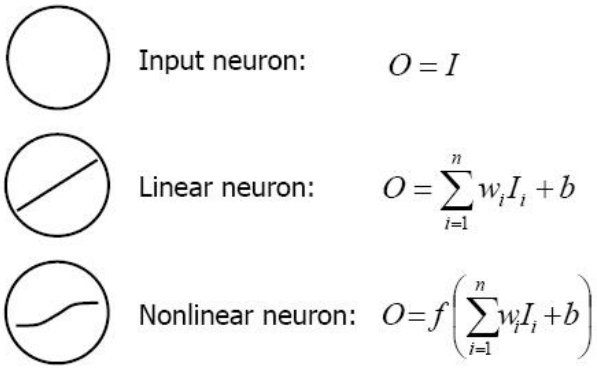


Fig. 2-11. Artificial neuron symbol definitions.

Generally, the p input variables are decomposed into nonlinear features by the n_m neurons in the mapping layer. These features are then combined and passed through the n_b bottleneck neurons, which perform a nonlinear transformation on the decomposed features. Next, the outputs of the bottleneck neurons are combined and reconstructed into nonlinear features by the n_m neurons in the demapping layer. These reconstructions are then linearly combined to form predictions for the p input variables by the output layer.

This process is more clearly illustrated by considering an example (see Fig. 2-12). Consider an AANN that has been developed to monitor four nonredundant sensors. Additionally, suppose that there is only one useful nonlinear relationship between the four sensor outputs. When query data for the four sensors are supplied to the AANN, they are first decomposed into nonlinear components by the mapping layer. These components are then combined and passed through a nonlinear transformation by the bottleneck layer. Since it was stated earlier that there is only one useful nonlinear relationship between the sensor outputs, it can be expected that one bottleneck neuron will result in a properly trained AANN. The additional nonlinear features, which may generally be attributed to random noise, are dropped from the parameter estimation process. The outputs of the bottleneck neurons are then reconstructed into nonlinear components, which are eventually combined into predictions for the four sensor values.

For additional information on AANNs and neural networks in general, the reader is referred to Haykin [1994].

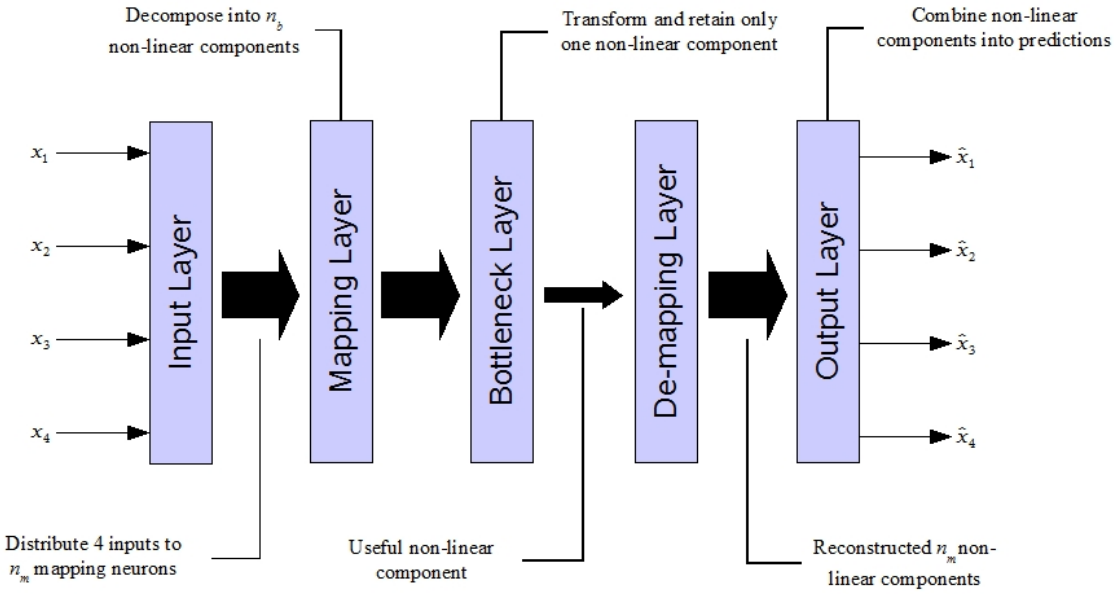


Fig. 2-12. Illustration of AANN signal propagation example.

2.2.6 ANN regularization techniques

As previously outlined in Sect. 2.3.1, inverse problems, such as those found in some OLM models, are often ill-posed. To better solve these problems numerically one must introduce some additional information about the solution, such as an assumption on the smoothness or a bound on the norm of the weights. This additional information will help prevent the model from being too complex for the system resulting solutions that have a lower variance and are more repeatable. When the model is overly complex, "overfitting" will occur and the model will learn the noise in the data rather than the true underlying functional relationship. In order to force the AANN complexity to match that of the system being modeled, three major regularization techniques are commonly employed [Gribok et al. 2002]. These include cross-validation training, controlled neuron growth, and neuron pruning, each of which will now be discussed in more detail.

2.2.6.1 Cross-validation training

Cross-validation training is a technique commonly used in neural networks to prevent over-training. Overtraining, a familiar phenomenon that occurs during neural network training, results when the neural

network complexity is higher than that of the relationship being modeled. As training progresses, an error criterion is iteratively minimized based on the training data; thus, the training error consistently decreases. Due to the presence of noise in the measured signals, if a similar but independent set of data (the validation data) is evaluated by the neural network after each iteration of training, the initial trend for the validation error will be to decrease; however, at some point, the validation error will begin to increase due to the training process learning the random noise in the training data that is different from that in the validation data. This phenomenon is referred to as overtraining. The process of cross-validation avoids overtraining by stopping the training process at the point where the validation error reaches a minimum and begins to increase. The model performance is quantified from an additional data set never used for training or validation, so optimal application of the cross-validation training need not be a concern. In other words, cross-validation training provides a method for restricting a neural network's capacity to fit noise even if there is a miss-specification between the complexity of the relationship and the neural network architecture complexity.

2.2.6.2 Controlled neuron growth and neuron pruning

The next two regularization techniques are very similar and require changing the number of neurons in a neural network to match the model's complexity to the complexity of the relationships in the data to be modeled. The techniques do this by either sequentially adding neurons until the network successfully trains and/or sequentially removing neurons from a previously trained network until it no longer successfully trains. Even though there is no universal process for determining the optimum number of neurons for a neural network, there are several "rules-of-thumb," especially for AANNs. The most significant of these is the observation that although changing the number of neurons in the mapping/demapping layer will affect AANN predictive performance, these effects are less significant than changes in the bottleneck layer. Therefore, as a general rule, it is usually more important to have a network with sufficient mapping/demapping neurons such that a minimal number of bottleneck neurons may be used, than it is to have increased bottleneck neurons for a reduced number of mapping/demapping neurons. In other words, the neurons in the bottleneck layer drive network complexity.

2.3 Auto-Associative Kernel Regression

2.3.1 Introduction

In statistics and empirical modeling, the process of estimating a parameter's value by calculating a weighted average of historical exemplar values is known as kernel regression [Atkeson 1997]. In general, kernel regression may be most compactly represented by the so-called Nardaraya [1964]-Watson [1964] estimator. Consider the simplest inferential model (i.e., one input $\boldsymbol{x}$, one output $\boldsymbol{y}$) whose Nardaraya-Watson estimator is given by the following equation:

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n}\left[\boldsymbol{K}_h\left(\boldsymbol{X}_i - \boldsymbol{x}\right)\boldsymbol{Y}\right]}{\sum_{i=1}^{n}\boldsymbol{K}_h\left(\boldsymbol{X}_i - \boldsymbol{x}\right)}, \tag{2.3.1.1}$$

where

$\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$	are exemplar predictor and response values respectively; $\boldsymbol{x}$ is a query predictor vector,
$\boldsymbol{K}_h\left(\boldsymbol{X}_i - \boldsymbol{x}\right)$	is a weighting function or kernel function, which generates a weight similarity) for a given difference of a query and exemplar vector,
$\hat{\boldsymbol{y}}(\boldsymbol{x})$	is an estimate of $\boldsymbol{y}$, given $\boldsymbol{x}$.

The nonparametric model operates by comparing a query $\boldsymbol{x}$ to past input examples $\boldsymbol{X_i}$. The output is a weighted average of past examples $\boldsymbol{Y_i}$. For example, if a query input is similar to stored inputs $\boldsymbol{X_{i=6,8,13}}$, then the output is a weighted average of output examples $\boldsymbol{Y_{i=6,8,13}}$. An alternative representation of Eq. (2.3.1.1) replaces the difference $\boldsymbol{X_1 - x}$ with a general distance function, $\boldsymbol{d_i(X_i, x)}$ as follows:

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n}\left\{\boldsymbol{K_h}\left[\boldsymbol{d_i}\left(\boldsymbol{X_i}, \boldsymbol{x}\right)\right]\boldsymbol{Y_i}\right\}}{\sum_{i=1}^{n}\boldsymbol{K_h}\left[\boldsymbol{d_i}\left(\boldsymbol{X_i}, \boldsymbol{x}\right)\right]} \quad . \tag{2.3.1.2}$$

2.3.2 Distance functions

A common distance function is the Euclidean distance, which is also known as the L^2-norm and is given by the following equation:

$$\boldsymbol{d_i}\left(\boldsymbol{X_i}, \boldsymbol{x}\right) = \sqrt{\left(\boldsymbol{X_i} = \boldsymbol{x}\right)^2} \quad . \tag{2.3.2.1}$$

More robust distance functions (i.e., erroneous data are less effective in degrading a parameter estimate) have also been investigated and include the L^1-norm [Hines and Garvey 2005]:

$$\boldsymbol{d_i}\left(\boldsymbol{X_i}, \boldsymbol{x}\right) = \left|\boldsymbol{X_i} - \boldsymbol{x}\right| \quad . \tag{2.3.2.2}$$

In the one dimensional case these are equivalent but will not be when there are multiple model inputs.

2.3.3 Kernel functions

Recall that the role of the kernel function is to generate a weight (similarity) for a given distance of a query vector from an exemplar vector. Therefore, it should have large values for small distances and small values for large distances. In other words, when a query vector is nearly identical to an exemplar, its distance should be small; therefore, that particular exemplar should receive a large weight. If a query vector did not resemble an exemplar, it should receive a small weight. One commonly used function that satisfies this criterion is the Gaussian kernel [Fan and Gijbels 1996]:

$$\boldsymbol{K_h}(\boldsymbol{d}) = \frac{1}{\sqrt{2\pi\boldsymbol{h}^2}}\boldsymbol{e}^{-\boldsymbol{d}^2/2\boldsymbol{h}^2} \quad . \tag{2.3.3.1}$$

Here, *h* is commonly referred to as the kernel's bandwidth and is used to control what effective distances are deemed similar. It can be seen in Fig. 2-13 that the Gaussian kernel with the smaller bandwidth ($\boldsymbol{h}$ = 0.1) will only generate large weights when the distance is very close to zero, while the kernel with the larger bandwidth is less specific and will assign significant weights for a larger range of distances.

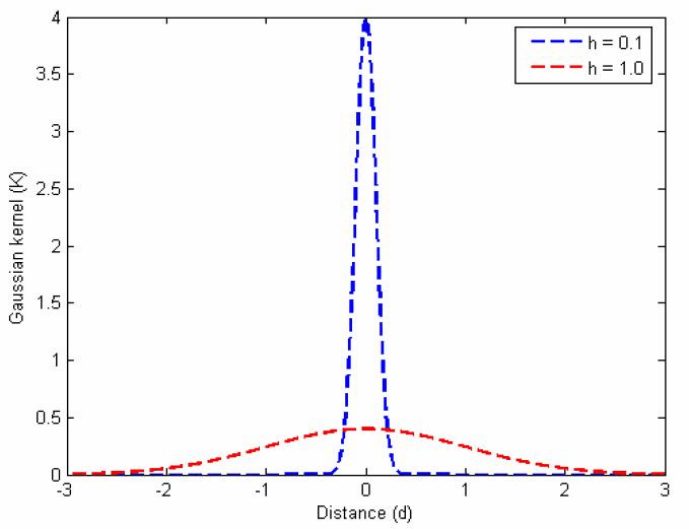


Fig. 2-13. Example of Gaussian kernels.

Other kernel functions include the inverse distance, exponential, absolute exponential, uniform weighting, triangular, biquadratic, and tricube kernels [Atkeson 1997], examples of which may be seen in Fig. 2-14. Although each function may have advantages in certain situations, the Gaussian kernel function is generally an adequate selection. In fact, the work of Scott [1992] and Cleveland and Loader [1994a, 1994b] show that the kernel function type plays a noncritical role in the performance of locally weighted models.

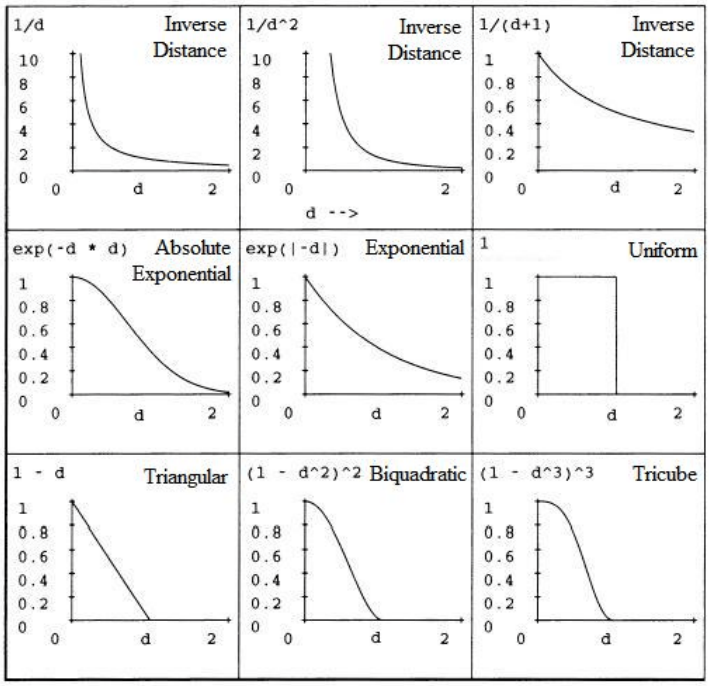


Fig. 2-14. Examples of alternative kernel functions.

2.3.4 Extending traditional kernel regression to autoassociation

In this section, a description of auto-associative kernel regression (AAKR) will be given. Because descriptions of AAKR do not yet appear in open literature, this derivation is based upon multivariate inferential kernel regression as derived by Wand and Jones [1995].

In this report, the exemplar or memory vectors used to develop the empirical model are represented by the matrix $\boldsymbol{X}$, where $\boldsymbol{X}_{i,j}$ is the ith observation of the jth variable. For $\boldsymbol{n_m}$ memory vectors and $\boldsymbol{p}$ process variables, this matrix becomes:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_{1,1} & \boldsymbol{X}_{1,2} & \cdots & \boldsymbol{X}_{1,p} \\ \boldsymbol{X}_{2,1} & \boldsymbol{X}_{2,2} & \cdots & \boldsymbol{X}_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{X}_{n_m,1} & \boldsymbol{X}_{n_m,2} & \cdots & \boldsymbol{X}_{n_m,p} \end{bmatrix} . \tag{2.3.4.1}$$

Using this format, a query vector is represented by the $1 \times \boldsymbol{p}$ matrix $\boldsymbol{x}$:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_p \end{bmatrix} . \tag{2.3.4.2}$$

The mathematical framework of this modeling technique is composed of three basic steps. First, the distance between a query vector and each of the memory vectors is computed. There are many different distance functions that may be used, but the most commonly used function is the Euclidean distance, for which the equation for the ith memory vector is as follows:

$$\boldsymbol{d}_i\left(\boldsymbol{X}_i, \boldsymbol{x}\right) = \sqrt{\left(\boldsymbol{X}_{i,1} - \boldsymbol{x}_1\right)^2 + \left(\boldsymbol{X}_{i,2} - \boldsymbol{x}_2\right)^2 + \ldots + \left(\boldsymbol{X}_{i,p} - \boldsymbol{x}_p\right)^2} . \tag{2.3.4.3}$$

For a single query vector, this calculation is repeated for each of the $\boldsymbol{n_m}$ memory vectors, resulting in an $\boldsymbol{n_m} \times 1$ matrix of distances $\boldsymbol{d}$**:**

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \vdots \\ \boldsymbol{d}_{n_m} \end{bmatrix} . \tag{2.3.4.4}$$

Next, these distances are used to determine weights by evaluating the Gaussian kernel, expressed by:

$$\mathrm{w} = \boldsymbol{K}_{\boldsymbol{h}}(\mathrm{d}) = \frac{1}{\sqrt{2\pi \boldsymbol{h}^2}} \boldsymbol{e}^{-\boldsymbol{d}^2 / \boldsymbol{h}^2} , \tag{2.3.4.5}$$

where

$\boldsymbol{h}$ is the kernel bandwidth; and

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n_m} \end{bmatrix}$$ are the n_m memory vector weights.

Finally, these weights are combined with the memory vectors to make predictions using the weighted average:

$$\hat{x} = \frac{\sum_{i=1}^{n_m} (w_1 X_i)}{\sum_{i=1}^{n_m} w_i} \quad . \tag{2.3.4.6}$$

If the scalar a is defined as the sum of the weights, that is,

$$a = \sum_{i=1}^{n_m} w_i \quad , \tag{2.3.4.7}$$

then Eq. (2.3.4.6) can be represented by the following more compact matrix form:

$$\hat{x} = \frac{w^T X}{a} \quad . \tag{2.3.4.8}$$

2.3.5 Regularization techniques

Regularization incorporates an a priori assumption that a model's input-output relationship should be smooth. Recall that the AAKR weights are generated by a kernel function, whose character may be changed by either increasing or decreasing its bandwidth. The question now arises, how does the character of the kernel function affect the smoothness of a model's input-output relationship? The answer to this question is easiest to understand by considering two extreme cases described earlier (see Fig. 2-13).

2.3.5.1 Kernel bandwidth and smoothness

First, consider a kernel function with a small bandwidth. Such a kernel function would only generate significant weights when the query vector is very near a memory vector. Therefore, it can be expected that for an arbitrary query vector, a very small number of memory vectors will influence its estimated value. In general, this represents a poor utilization of information because a model's generalization capabilities (i.e., production of valid estimates for parameters that lie between training states) are inherently a function of how much information is used to estimate its value. AAKR models that use small kernel bandwidths can be expected to produce rough input-output relationships as the weights will generally oscillate between individual memory vectors.

Next, consider a kernel function with a large bandwidth. Such a kernel function would generate many significant weights, because it assigns significant weights to a larger range of distance values. Following a similar argument used earlier, for an arbitrary query vector a relatively large number of

memory vectors will influence its estimated value. Such a kernel function would then act to smooth the AAKR's input-output relationship, because more memory vectors are used in the weighted average.

A comparison of an example AAKR model's predictions for small and large bandwidths is presented in Fig. 2-15. Notice that the predictions of the AAKR with the smaller bandwidth are significantly rougher than those of the larger bandwidth.

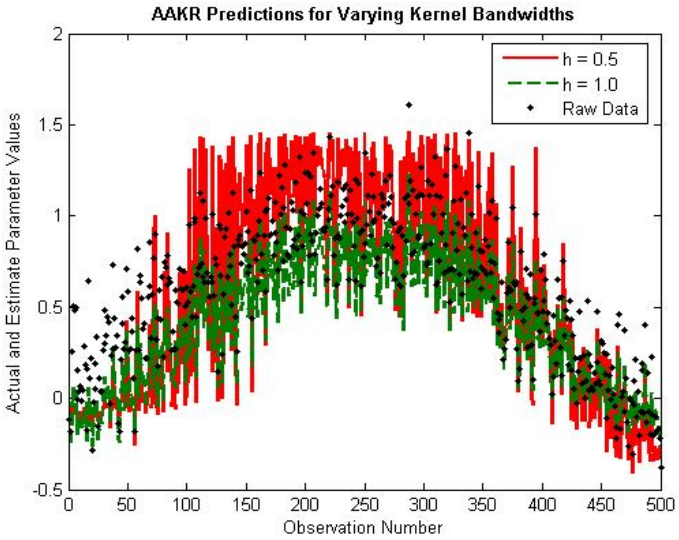


Fig. 2-15. Illustration of the relationship between AARK bandwith and the smoothness of parameter estimates.

From these cases, a general relationship between kernel bandwidth and "smoothness" is found to be as follows:

- Small bandwidths generally produce rough predictions.
- Large bandwidths generally produce smooth predictions.

At first glance it is tempting to jump to the conclusion that since regularization produces a smooth input-output relationship, then a large bandwidth should always be used. This conclusion, however, is incorrect in that as the bandwidth increases, the model predictions will approach the mean value of the memory vectors, thus biasing the solution. For example, if an AAKR model was trained to generate a sine wave, the model's estimates will approach zero, regardless of the character of the query vector, for increasingly large bandwidths. Such a model would have no practical value, and therefore a methodology is needed for locating the bandwidth which provides the best trade-off between rough and smooth parameter estimates. Generally, bandwidth optimization proceeds as follows:

1. Define n test bandwidth values and set the iteration index $\boldsymbol{i}$ to 1.
2. Construct a prototype model with the ith kernel bandwidth.
3. Score the model's performance according to an objective function.
4. If $\boldsymbol{i} \neq \boldsymbol{n}$, go to (a), otherwise go to (b).
   (a) Store model score in a history variable or file, increment $\boldsymbol{i}$, and return to step 2.
   (b) Identify the minimum objective function score from the optimization history and label its corresponding bandwidth as having the optimal value.

Notice in step 4(b) that the optimum bandwidth is that which minimizes an objective function. A minimum value is used instead of the maximum in this generic description because the objective function is usually an error measure. In the following sections, two scoring methods that are used in bandwidth optimization are discussed.

2.3.5.2 Cross-validation error minimization

The first scoring method is cross-validation error minimization. In this method, memory vectors and a kernel bandwidth are selected to create a prototype model. Next, the prototype model is used to estimate the values of n_v test observations that contain feature patterns not present in the training data. Finally, the test data are subtracted from the prototype model estimates and used to calculate the mean squared error (score).

For an auto-associative model, calculating the mean squared error (MSE) involves calculating the MSE for each modeled variable and then calculating the mean of these values. Here $\boldsymbol{e}$ is defined as a $1 \times \boldsymbol{p}$ matrix of mean squared errors for the $\boldsymbol{p}$ variables, where $\boldsymbol{e_j}$ is the MSE for variable $\boldsymbol{j}$,

$$\boldsymbol{e}_j = \sum_{\boldsymbol{k}=1}^{\boldsymbol{n_v}} \left[\hat{\boldsymbol{x}}_{\boldsymbol{k},\boldsymbol{j}} - \boldsymbol{x}_{\boldsymbol{k},\boldsymbol{j}} \right]^2 \quad , \tag{2.3.5.1}$$

where, $\boldsymbol{x_{k,j}}$ refers to the kth test observation of variable $\boldsymbol{j}$. The score for the ith prototype model is found by calculating the mean value of $\boldsymbol{e}$, as follows.

$$\boldsymbol{S}_i = \frac{1}{\boldsymbol{p}} \sum_{j=1}^{\boldsymbol{p}} \boldsymbol{e}_j \quad . \tag{2.3.5.2}$$

Because the patterns that are present in the test data have not been used to define the model, the model which minimizes $\boldsymbol{S_i}$ is the model that can be expected to produce the best estimates for future, unseen patterns.

2.3.5.3 Cross-validation uncertainty minimization

The second scoring method is called cross-validation uncertainty minimization and is equivalent to cross-validation error minimization with the exception that an estimate of the model's uncertainty is used to calculate a score instead of its prediction error. For a complete discussion of methods for estimating uncertainty, see Chaps. 5 and 6 of this report. For the time being, it is assumed that methods exist for estimating a model's uncertainty and that $\boldsymbol{u}$ is a $1 \times \boldsymbol{p}$ matrix of mean uncertainties for each model variable. The score for the ith prototype model is the mean value of $\boldsymbol{u}$ is

$$\boldsymbol{S}_i = \frac{1}{\boldsymbol{p}} \sum_{j=1}^{\boldsymbol{p}} \boldsymbol{u}_j \quad . \tag{2.3.5.3}$$

2.4 Auto-Associative Multivariate State Estimation Technique

2.4.1 Introduction

The Multivariate State Estimation Technique (MSET) is a nonlinear, nonparametric modeling technique that was developed by Argonne National Laboratory (ANL) [Gross et al. 1998, 2001; Singer et al. 2001] for high-sensitivity proactive fault monitoring applications in advanced nuclear power systems. In addition to nuclear power plants, MSET has been used in a wide variety of monitoring systems that include (1) NASA's realtime signal validation, sensor operability validation, and proactive fault monitoring for the space shuttle vehicle and ground support systems [Bickford et al. 2000, 2001]; and (2) Sun Microsystems' proactive maintenance software for enterprise class computing hardware [Gross et al. 2002a, 2002b, 2002c]. MSET is very similar to AAKR and only differs by the normalization method. The innovative derivation of MSET is given by Singer et al. [2001], which is outlined below.

If $\boldsymbol{n_m}$ exemplar vectors of p sensor values are collected from a system, then the process memory $\boldsymbol{D}$, commonly referred to as the memory matrix, may be represented as follows.

$$\boldsymbol{D} = \boldsymbol{X}^{\boldsymbol{T}} = \begin{bmatrix} \boldsymbol{X}_{1,1} & \boldsymbol{X}_{2,1} & \cdots & \boldsymbol{X}_{n_m,1} \\ \boldsymbol{X}_{2,1} & \boldsymbol{X}_{2,2} & \cdots & \boldsymbol{X}_{n_m,2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{X}_{1,p} & \boldsymbol{X}_{2,p} & \cdots & \boldsymbol{X}_{n_m,p} \end{bmatrix} . \tag{2.4.1.1}$$

The memory matrix $\boldsymbol{D}$ may now be used to estimate the values for a query observation of the $\boldsymbol{p}$ system variables, denoted by the $1 \times \boldsymbol{p}$ vector x. An estimate of $\boldsymbol{x}$ may be calculated with:

$$\hat{\boldsymbol{x}}^{\boldsymbol{T}} = \boldsymbol{D}\left(\boldsymbol{D}^{\boldsymbol{T}}\boldsymbol{D}\right)^{-1}\boldsymbol{D}^{\boldsymbol{T}}\boldsymbol{x}^{\boldsymbol{T}} \ . \tag{2.4.1.2}$$

This linear solution minimizes the Euclidean norm between the estimated and measured data vectors. This result has several clear limitations, specifically the requirement that $\boldsymbol{D}^{\boldsymbol{T}}\boldsymbol{D}$ be nonsingular, an inability to accommodate random uncertainties, and nonrandom defects. MSET addresses these limitations by introducing a nonlinear operator, denoted by ⊗, into Eq. (2.4.1.2):

$$\hat{\boldsymbol{x}}^{\boldsymbol{T}} = \boldsymbol{D}\left(\boldsymbol{D}^{\boldsymbol{T}} \otimes \boldsymbol{D}\right)^{-1}\left(\boldsymbol{D}^{\boldsymbol{T}} \otimes \boldsymbol{x}^{\boldsymbol{T}}\right) \ . \tag{2.4.1.3}$$

Singer et al. [2001] describe a suitable nonlinear operator as having the following properties:

1. $\boldsymbol{D}^{\boldsymbol{T}} \otimes \boldsymbol{D}$ must be nonsingular;
2. if some elements in the observation vector are not within the ranges of the same elements of the column vectors in the memory matrix $\boldsymbol{D}$, the estimation vector must still represent an optimum estimation;
3. if the observation vector is identical to one of the column vectors in $\boldsymbol{D}$, then the estimation vector must be identical to the observation vector; and
4 the error vector (difference between the observation and estimation vectors) must be minimized.

Two such operators that fulfill these conditions have been identified but are not available in the literature due to patent and proprietary issues. In the next section, Eq. (2.4.1.3) will be manipulated, such that its final form will be very near that of kernel regression. Furthermore, it will be shown that the MSET nonlinear operator is a type of kernel function and therefore, should not provide critical improvements [Scott 1992; Cleveland and Loader 1994a, 1994b] to parameter estimates as compared to the widely used Gaussian kernel [Wolpert 1997]. Researchers at the University of Tennessee have already investigated the operators used in the commercially available MSET and determined that they perform similarly to the Gaussian kernel. Additional testing has been performed that supports the findings that no one operator is better than all other operators in a general OLM system. It is expected that a TS license amendment would disclose the kernel type and would also need to demonstrate its proficiency in this algorithm.

2.4.2 MSET vs kernel regression

Recall that the estimation equation of AAKR for an input vector $\boldsymbol{x}$ is

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{w}^T \boldsymbol{X}}{\boldsymbol{a}} \ . \tag{2.4.2.1}$$

Also, recall that $\boldsymbol{w}$ is an $\boldsymbol{n_m} \times 1$ matrix of weights obtained by evaluating the Gaussian kernel with the distances of $\boldsymbol{x}$ from each of the $\boldsymbol{n_m}$ exemplar vectors contained in $\boldsymbol{X}$, and $\boldsymbol{a}$ is a normalizing constant (sum of the weights).

2.4.2.1 MSET function manipulation

To begin the comparison, recall that the memory matrix $\boldsymbol{D}$ is the transpose of $\boldsymbol{X}$. Making this substitution yields:

$$\hat{\boldsymbol{x}}^T = \boldsymbol{X}^T \left[\left(\boldsymbol{X}^T \right)^T \otimes \boldsymbol{X}^T \right]^{-1} \left[\left(\boldsymbol{X}^T \right) \otimes \boldsymbol{x}^T \right] \ . \tag{2.4.2.3}$$

Canceling the appropriate transpose operations:

$$\hat{\boldsymbol{x}}^T = \boldsymbol{X}^T \left[\boldsymbol{X} \otimes \boldsymbol{X}^T \right]^{-1} \left[\boldsymbol{X} \otimes \boldsymbol{x}^T \right] \ . \tag{2.4.2.4}$$

Next, taking the transpose of both sides:

$$\hat{\boldsymbol{x}} = \left[\boldsymbol{x} \otimes \boldsymbol{X}^T \right] \left[\boldsymbol{X} \otimes \boldsymbol{X}^T \right]^{-1} \boldsymbol{X} \ . \tag{2.4.2.5}$$

Notice, that there is a significant change in operation (i.e., $\otimes$ and multiplication) order. This change is a result of the following matrix property:

$$\left(\boldsymbol{AB} \right)^T = \boldsymbol{B}^T \boldsymbol{A}^T \ . \tag{2.4.2.6}$$

Also, notice that $\left[\boldsymbol{X} \otimes \boldsymbol{X}^T \right]^{-1}$ remained unchanged from Eq. (2.4.2.4) to Eq. (2.4.2.5). This is a result of the following properties:

$$\left(\boldsymbol{A}^{-1}\right)^{\boldsymbol{T}} = \left(\boldsymbol{A}^{\boldsymbol{T}}\right)^{-1} \Rightarrow \left[\left(\boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}}\right)^{-1}\right]^{\boldsymbol{T}} = \left[\left(\boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}}\right)^{\boldsymbol{T}}\right]^{-1} \tag{2.4.2.7}$$

$$\left(\boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}}\right)^{\boldsymbol{T}} = \boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}} \ . \tag{2.4.2.8}$$

The properly represented by Eq. (2.4.2.8) implies that $\boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}}$ is a symmetric matrix (i.e., $\boldsymbol{A}^{\boldsymbol{T}} = \boldsymbol{A}$). This properly will be illustrated in the next section, which describes how the nonlinear operator ⊗ is applied.

2.4.2.2 Applying the nonlinear operator

To further condense Eq. (2.4.2.5), the process of applying the nonlinear operator ⊗ must be understood. First, consider, $\boldsymbol{x} \otimes \boldsymbol{X}^{\boldsymbol{T}}$, which is fully represented by the following matrices:

$$\boldsymbol{x} \otimes \boldsymbol{X}^{\boldsymbol{T}} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_p \end{bmatrix} \otimes \begin{bmatrix} \boldsymbol{X}_{1,1} & \boldsymbol{X}_{2,1} & \cdots & \boldsymbol{X}_{\boldsymbol{n}_m,1} \\ \boldsymbol{X}_{1,2} & \boldsymbol{X}_{2,2} & \cdots & \boldsymbol{X}_{\boldsymbol{n}_m,2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{X}_{1,\boldsymbol{p}} & \boldsymbol{X}_{2,\boldsymbol{p}} & \cdots & \boldsymbol{X}_{\boldsymbol{n}_m,\boldsymbol{p}} \end{bmatrix} . \tag{2.4.2.9}$$

As with matrix multiplication, the application of ⊗ produces a $1 \times \boldsymbol{n}_m$ matrix, as verified by the following equation:

$$(1 \times \boldsymbol{p})(\boldsymbol{p} \times \boldsymbol{n}_m) = 1 \times \boldsymbol{n}_m \ . \tag{2.4.2.10}$$

The process by which this calculation is performed proceeds according to the following steps (see Fig. 2-16):

1. Set $\boldsymbol{i} = 1$.
2. Extract the ith column from $\boldsymbol{X}^T$, which is the ith memory vector $\boldsymbol{X}_i$.
3. Calculate the distance of $\boldsymbol{x}$ from $\boldsymbol{X}_i$.
4. Supply the distance to the MSET kernel and record its output in the ith column of weight matrix $\boldsymbol{w}$.
5. If $\boldsymbol{i} \neq \boldsymbol{n}_m$, go to (a), otherwise go to (b).
   (a) Increment $\boldsymbol{i}$ and return to step 2.
   (b) Stop.

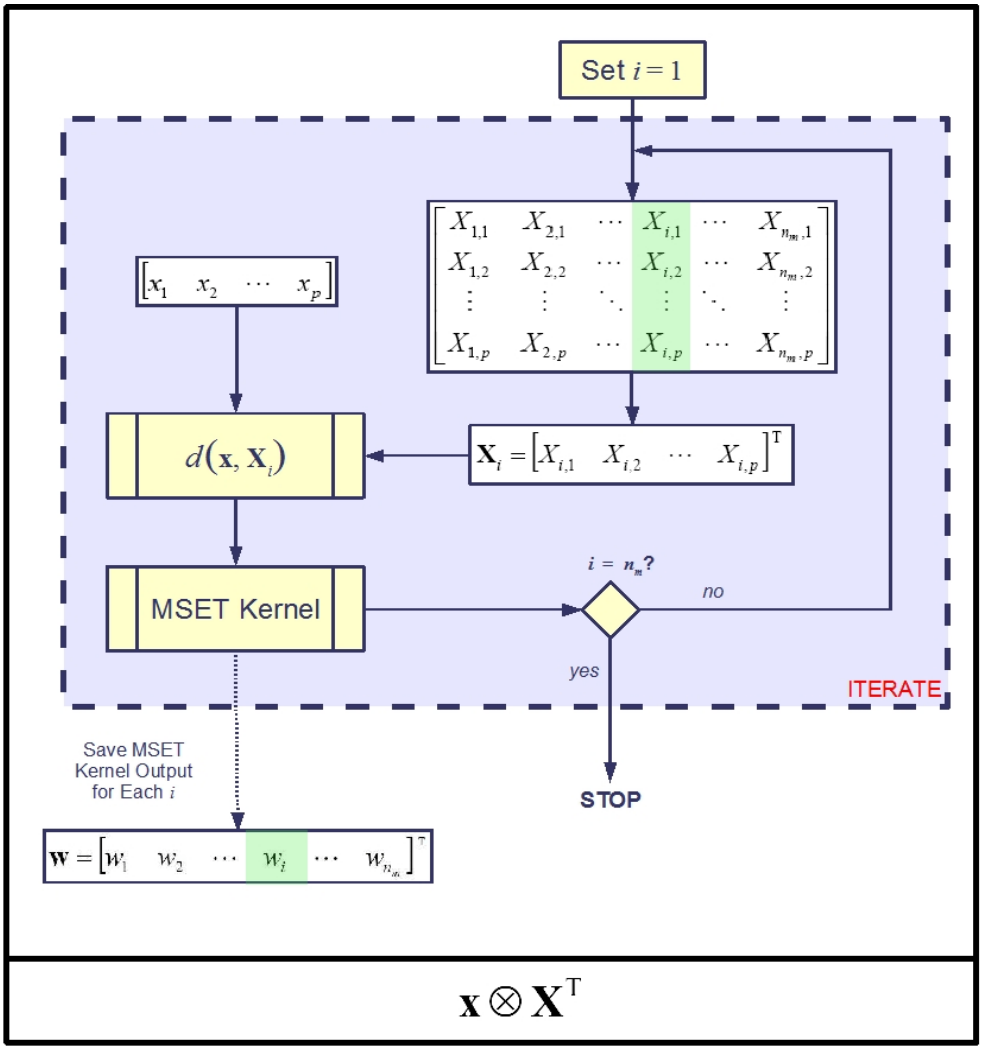


Fig. 2-16. Process diagram for applying the MSET nonlinear operator for a *x* and *X*.

Notice that when the Gaussian kernel function is used, that ***w*** is identical to the ***w*** used in kernel regression.

Next, consider $\boldsymbol{X} \otimes \boldsymbol{X}^T$, which is calculated according to the following steps:

1. Set i = 1 and $\boldsymbol{j} = 1$.
2. Extract the *i*th row from ***X***, which is the *i*th memory vector $\boldsymbol{X_i}$.
3. Extract the *j*th column from X^T, which is the *j*th memory vector $\boldsymbol{X_j}$.
4. Calculate the distance of $\boldsymbol{X_i}$ from $\boldsymbol{X_j}$.
5. Supply the distance to the MSET kernel, and record its output in the *i*th row and *j*th column of a normalizing matrix **N**.
6. Evaluate the appropriate iteration case.
   (a) If $\boldsymbol{i} \neq \boldsymbol{n_m}$ and $\boldsymbol{j} \neq \boldsymbol{n_m}$, then set $\boldsymbol{j} = \boldsymbol{j} + 1$ and return to step 3.

(b) If $i \neq n_m$ and $j = n_m$, then set $i = i + 1$, $j = 1$, and return to step 2.

(c) If $i = n_m$ and $j = n_m$, then stop.

Examining the above process, three major features of the normalizing matrix $\boldsymbol{N}$ are evident. First, the dimensions of $\boldsymbol{N}$ are $n_m \times n_m$. Next, the diagonal elements of $\boldsymbol{N}$ are simply the kernel function's maximum value, since

$$N_{i,i} = K_h\left[d\left(X_i, X_i\right) = 0\right] . \tag{2.4.2.11}$$

Finally, since distance functions do not generally depend on order, that is:

$$d\left(X_i, X_j\right) = d\left(X_j, X_i\right) , \tag{2.4.2.12}$$

then the elements of $\boldsymbol{N}$ are symmetric.

$$N_{i,j} = N_{j,i} . \tag{2.4.2.13}$$

This feature verifies the symmetry property presented earlier in Eq. (2.4.2.8).

2.4.2.3 MSET normalization and effective kernel

The MSET prediction equation is now given by:

$$\hat{x} = w^T N^{-1} X . \tag{2.4.2.14}$$

Rearranging Eq. (2.4.2.14), an equation very similar to kernel regression is finally obtained:

$$\hat{x} = \frac{w^T X}{N} . \tag{2.4.2.15}$$

It can be seen that the normalizing matrix $\boldsymbol{N}$ performs a similar task of normalizing a model's estimates as a in kernel regression. The difference lies in the fact that N^{-1} acts to change the Gaussian kernel into a higher order kernel. An example of this transformation is illustrated in Fig. 2-17, which is a plot of ordered MSET weights for a query vector located near the middle of a memory vector range. In summary, MSET is equivalent to AAKR with a higher-order kernel that provides additional flexibility and complexity. It is not apparent that this higher complexity provides any advantages for OLM tasks.

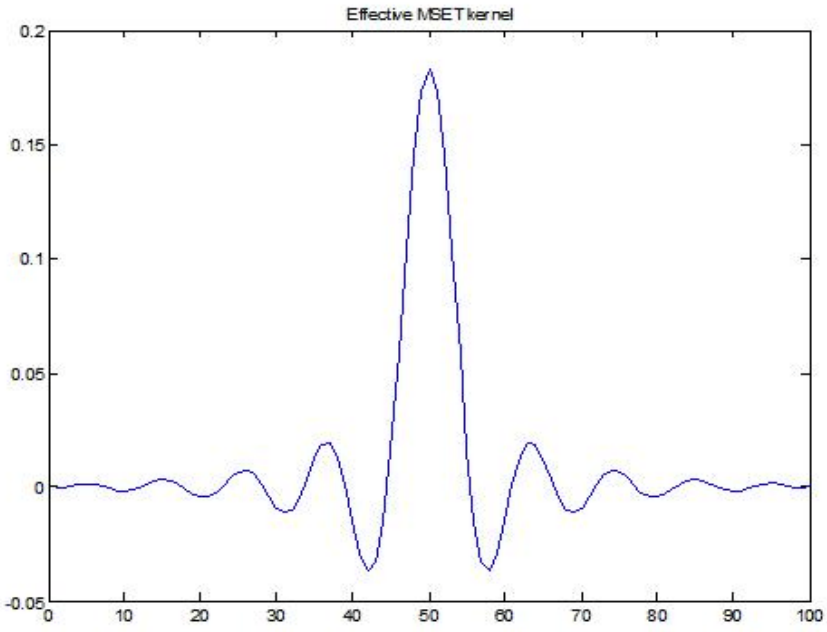


Fig. 2-17. Illustration of effective MSET kernel.

2.4.3 Regularization techniques

Because auto-associative MSET (AAMSET) is very similar to AAKR, the regularization methods discussed in Sect. 2.3.5 are applicable to AAMSET. There are, however, two additional regularization techniques that may be used with AAMSET.

2.4.3.1 Vector selection

Recall that AAMSET requires the inversion of a normalizing matrix $\boldsymbol{N}$, which is defined as:

$$\boldsymbol{N} = \boldsymbol{X} \otimes \boldsymbol{X}^{T} \quad . \tag{2.4.3.1}$$

Predictions are made by comparing a query observation with historical observations called memory vectors. Because a query vector must be compared to each memory vector, large training data sets can easily lead to unacceptable computational loads. Although counterintuitive, smaller training sets may be needed because the solution to most OLM models use the inversion of the D'⊗D matrix, which is ill-posed and unstable when memory vectors are similar. This is similar to ill-conditioned $\boldsymbol{X}'\boldsymbol{X}$ matrices in linear regression that occur when the inputs are correlated (somewhat dependent). Consider the fact that if the number of training observations is large compared to the number of operating states of the system being modeled, then the number of similar rows in $\boldsymbol{X}$ would be large. Because the nonlinear operator ⊗, as used in Eq. (2.4.3.1), effectively measures the similarity between the rows of $\boldsymbol{X}$, $\boldsymbol{N}$ will be poorly conditioned, which results from the repeating of patterns (similarities) in $\boldsymbol{N}$. The end result of this situation is that the AAMSET model will be unstable and produce unreliable parameter estimates.

One solution to this conditioning problem is to intelligently select a reduced set of memory vectors from the training data and then use the selected vectors to develop the AAMSET model. One such vector selection algorithm is described by Gross et al. [2002a] and involves the following two steps: (1) the

observations containing the minimum and maximum value for each included signal are selected, and (2) additional vectors are selected by first ordering the training data observations according to their Euclidean norm and then selecting observations at equal intervals to fill the AAMSET model with a user-specified number of operating examples. A typical number of memory vectors for MSET is on the order of 50 while AAKR may use 500. Memory vector selection techniques are discussed in more detail in Volume 3 of this series.

2.4.3.2 Ridge regularization

Another method of AAMSET regularization involves implementing a form of ridge regularization to $\boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}}$ [Hines and Usynin 2005]. In a similar manner as ridge regression [Hoerl and Kennard 1970], a regularization parameter λ is applied by:

$$\boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}} + \lambda \boldsymbol{I} \ . \tag{2.4.3.2}$$

Here, $\boldsymbol{I}$ is an identity matrix having the same dimension as $\boldsymbol{X} \otimes \boldsymbol{X}^{\boldsymbol{T}}$. It has been shown by Hoerl and Kennard [1970] that a decrease in the condition number (ratio of the maximum and minimum Eigen value) of a matrix greatly improves the prediction stability of a model developed from its inversion. In Eq. (2.4.4.2), λ dampens the effects of the smaller Eigen values, which increases the effective minimum Eigen value and improves the condition of the matrix to be inverted. This produces more stable and repeatable predictions.

2.5 Performance Measures

The performance of auto-associative OLM systems is measured in terms of three metrics: accuracy, auto-sensitivity, and cross-sensitivity. Accuracy measures the ability of a model to correctly and accurately predict sensor values and is normally presented as the MSE between the predictions and the measured sensor values. Auto-sensitivity measures a model's ability to make correct sensor predictions when the respective sensor value is incorrect due to some sort of fault. Finally, cross-sensitivity measures the effect a faulty sensor input has on the predictions for other sensor predictions of the model. For additional information refer to the work of Hines and Usynin [2004]. Two newer metrics are also discussed in this section: Error Uncertainty Limit Monitoring (EULM) detectability, and Sequential Probability Ration Test (SPRT) detectability, which have been recently introduced for quick validation of OLM applicability for sensor calibration monitoring [Hines and Garvey 2006]. EULM and SPRT detectability quantify the smallest sensor calibration fault and anomaly that may be identified by an empirical model, respectively. Because an ideal model would be accurate, would have sensor predictions that are not appreciably affected by degraded inputs, and would be able to detect small sensor faults and anomalies, all of these metrics become extremely important to OLM models.

Before discussing these metrics in more detail, a summary of the general testing process used in these performance calculations is given. To begin, a model's response using unfaulted input data is calculated and used to determine the accuracy metric. These data are commonly termed "test data" or "validation data" and should be data not used for model training or optimization. Next, each of the input variables is sequentially artificially drifted and applied to the models. The predictions using faulty input data are then used to determine the model's auto and cross-sensitivity. Each of the three metrics is now discussed in more detail, starting with the accuracy metric.

2.5.1 Accuracy

The accuracy metric is simply defined as the MSE between the model's predictions and the target values. It is important to note that this metric compares the model predictions with the sensor

measurements (inputs) for data that were not used for model training or optimization. The equation for a single variable is simply:

$$A = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{x}_i - x_i\right)^2 \quad . \tag{2.5.1.1}$$

where

N is the number of test observations,

$\hat{x}_i$ is the model prediction of the ith test observation,

x_i is the ith observation of the test data.

Although this metric is termed accuracy, it is actually a measure of error, and a low value is desired. The accuracy metric is the most commonly cited metric because it represents a model's performance for unfaulted input data. However, since the purpose of empirical modeling in OLM systems is to identify sensor and process faults, model performance under faulted conditions is important and must be quantified.

2.5.2 Sensitivity

As described by Gribok et al. [2002], the original concept of robustness can be traced to inferential regression models. In this context, a robust model is defined as a model that produces little to no changes in its output for small errors in its inputs. Extending this idea to auto-associative empirical models, a robust model would produce little to no changes in all of its outputs for errors in each of its inputs. This concept was first used to quantify OLM modeling performance by Uhrig et al. [1996] and Wrest et al. [1996] in which sensitivity measures were defined. Similarly, Wegerich [2002] developed equations for performance termed robustness and spillover, for which we define as auto-sensitivity and cross-sensitively, respectively.

On-line modeling uses empirical models to predict the true values of process parameters that are measured by sensors and their associated instrumentation. The predictions are compared to the measured values, and the differences are called residuals. Under normal conditions, the residuals are attributed to noise and are small random values that vary around zero. When residuals start to grow, an abnormal event is occurring. A model prediction needs to give the correct value even when sensor measurements that are supplied to the model contain errors. If a model prediction drifted along with a drifting sensor, the difference would be zero and the fault would be masked. Therefore, a model that is insensitive to input faults is desired.

To calculate the sensitivity of a prediction, first a model's response using fault-free input data is calculated. It is important that a data set which covers the entire operating region be chosen because the sensitivity could be different at different points. Next, each input variable is sequentially artificially faulted, and the model outputs are recorded. These predictions using faulty input data are then used to determine the model's sensitivity metrics.

Model sensitivity is generally defined as a measure of the change in the prediction of the ith variable ($\hat{x}_i$) produced by a change in its respective input (x_i):

$$S_i = \frac{\Delta \hat{x}_i}{\Delta x_i} \quad . \tag{2.5.2.1}$$

Auto-sensitivity (S_A) is a measure of an empirical model's ability to make correct sensor predictions when it's respective input sensor value is incorrect due to some sort of fault. Therefore, this metric involves the following values: sensor i's prediction with no fault in the input $\hat{x}_i$, sensor i's prediction with a faulted input $\hat{x}_i^{drift}$, sensor i's unfaulted input value x_i, and sensor i's drifted input value x_i^{drift}. A histogram of the measurements should then be used to determine the variation between the individual sensitivity estimates. If there is little or no variation, then a direct average over the operating region defined by N samples can be computed. Using these definitions, the auto-sensitivity for sensor i is found with the following equation:

$$S_{A,i} = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{\hat{x}_{ki}^{drift} - \hat{x}_{ki}}{x_{ki}^{druft} - x_{ki}}\right| . \qquad (2.5.2.2)$$

However, if the histogram shows that the average is not suitable, then only the worst 5% or 10% of the measurements should be included in the average to give a conservative "worst-case scenario" estimate of the auto-sensitivity.

If a model's auto-sensitivity is one, then the model's prediction follows the fault, resulting in a residual of zero, and the fault cannot be detected. If a model's auto-sensitivity value is between zero and one, which is the common case, the residual will underestimate the size of the sensor fault, and the OLM system drift limits may need to be compensated accordingly.

The next performance metric is cross-sensitivity. This value measures the effect a faulty sensor input (i) has on the predictions of sensor (j). This is illustrated by the following equation, in which j is the index of the unfaulted variable whose spillover metric is being calculated:

$$S_{C,ij} = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{\hat{x}_{kj}^{drift} - \hat{x}_{kj}}{x_{kj}^{drift} - x_{kj}}\right| \quad \text{for } i \neq j . \qquad (2.5.2.3)$$

Like the auto-sensitivity metric, a histogram of the individual measurements should be used with the cross-sensitivity metric to show that the average is suitable. If the average is not acceptable, then only the worst 5% or 10% of the values should be included in the calculation of the cross-sensitivity.

To more clearly illustrate the concept of sensitivity, consider the plots presented in Fig. 2-18. Two plots are included to illustrate the differences between an empirical model with small (upper plot) and large (lower plot) sensitivity metrics. In both plots, the points indicate normal, undrifted sensor data; the line drifting to the bottom at the highest rate indicates artificially drifted sensor value that is supplied as an input to the empirical model, and the upper line represents the model predictions of the correct sensor values. Notice that the model predictions with a small sensitivity (upper plot) lie very near the normal, undrifted data. This indicates that the model is not significantly affected by the drifted input and is able to accurately predict the parameter's actual value when supplied with faulty input. This model is considered to be a "robust" model. Next, consider the plot for the model with a large sensitivity metric (lower plot). Notice that the model's predictions lie very near the artificially drifted values. Such a model would be of little use in instrument calibration validation because its predictions follow the faulted inputs rather than correcting them. The model would need to be reoptimized or retrained, or an entirely new modeling technique would need to be employed before implementing the model into an OLM system.

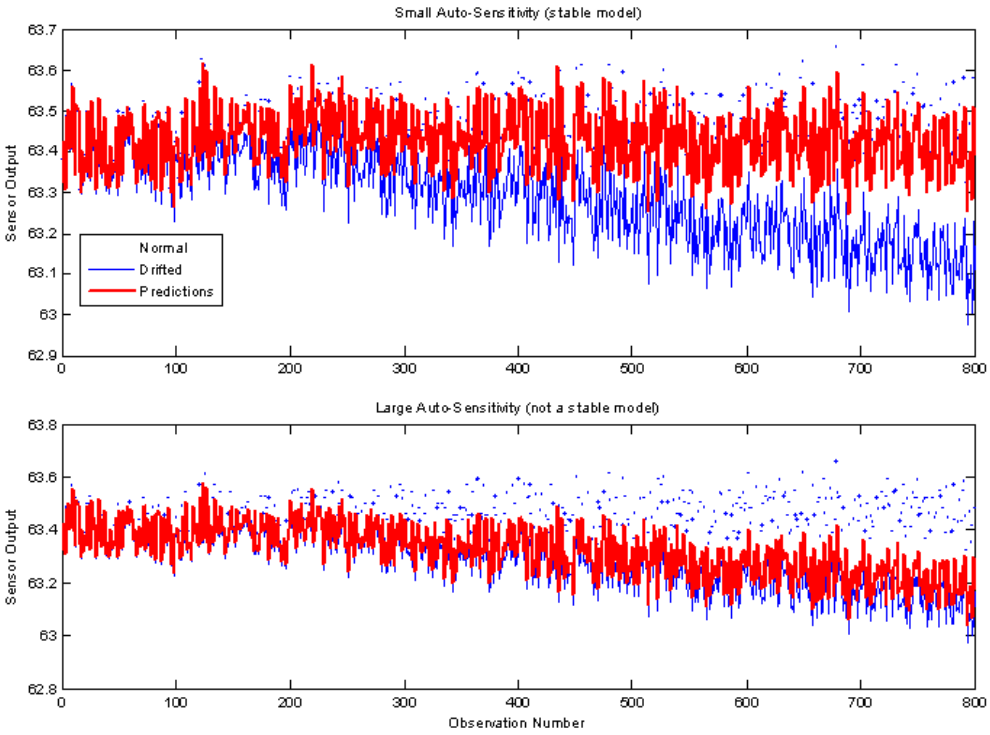


Fig. 2-18. Illustration of sensitivity performance metric.

Notice that the accuracy metric is calculated for each variable, the auto-sensitivity metric is calculated for the artificially drifted variable, and the cross-sensitivity metric is calculated for each variable except the artificially drifted variable. Consider a model with five variables. If a single variable is drifted, then there are five accuracy, one auto-sensitivity, and four nonzero cross-sensitivity metric values. To quantify the sensitivity values for each variable, an algorithm is structured so that each variable is sequentially perturbed. The auto-sensitivity metric for each variable is calculated, and the mean or 95% confident spillover metric for each variable is calculated.

2.6 Fault Detectability

All empirical predictive models used for OLM have uncertainty associated with their predictions. This uncertainty is caused by a multitude of factors including sensor noise, training data selection, model specification, model development, regularization [Gribok et al. 2002] and others. The uncertainty inherent in the model predictions affects the size of the fault that can be confidently detected. If a model prediction has 2% uncertainty defined by some sort of 95% confidence interval, then sensor drifts of 1% cannot be successfully identified. Therefore, it is important to develop models with minimal uncertainty, to quantify the uncertainty [Hines and Rasmussen 2005], and to properly consider the predictive uncertainty when making decisions.

Consider Fig. 2-19 that is a plot of a residual (the difference between an OLM model's estimate and the actual measured value) with its 95% confidence interval as a dotted line. The drift detection limits are set relative to the instrument channel uncertainties used in the set point analysis and the single-point monitoring uncertainty introduced by OLM [EPRI 2004]. The drift detection limits—presented as "Drift Limits" in the figure—must not be confused with the "Deviation Limit" as used in extensive discussions with industry in regard to Set Point-Related Technical Specifications. The drift limits used here refer the conservative limits that are used to identify the onset of a drift problem. If the residual reaches the limit,

then the instrument must be investigated. The drift limits used here must include factors other than drift, such as the effects of variations in the normal operating environment. But consideration is limited here to the sensors and those instruments involved in bringing the signals to the OLM computer, whereas the deviation limits are applied only to the instruments involved in periodic testing, which usually excludes the sensor. The analysis requires a 95% confidence that the sensor value has not exceeded its drift limits. When the 95% confidence interval (CI) crosses the drift limit, the channel would be flagged as needing investigation, and depending on the plant's procedures and the extent the CI exceeds the drift limits, the channel may be declared inoperable until the situation is resolved. If the uncertainty interval is larger than the drift limit, OLM cannot be used to monitor the channel because there is no allowance for drift. It can be seen that a small uncertainty interval allows a greater margin for drift before the channel is flagged. A more restrictive drift threshold may be specified that would give earlier warning to sensor performance degradation.

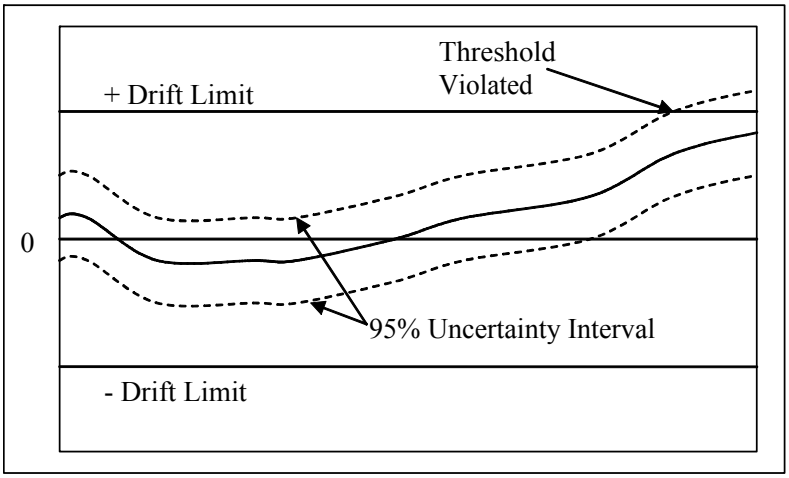


Fig. 2-19. On-line sensor calibration monitoring diagram.

Two fault detectability performance metrics will be presented in the next two sections. The first considers both the model auto-sensitivity and predictive uncertainty to quantify detectability for instrument calibration monitoring, while the second investigates a process to find the maximal sensitivity setting for a sequential probability ratio test (SPRT) used for anomaly detection. Although only the first is applicable to safety critical calibration monitoring, both are presented because the SPRT has been used extensively for OLM.

2.6.1 Error uncertainty limit monitoring (EULM)

In addition to uncertainty, there are two more restrictions on the drift detection level. First, when a sensor is calibrated it must be left with some predefined calibration tolerance. The set point analysis is used to determine the trip set point that will protect the reactor during a severe accident. This analysis considers assumed uncertainties inherent in the instrument channel. A manual calibration check assures the sensor channel is operating within prespecified assumptions and leaves the channel within the calibration tolerance. Drift is assumed to be time based, and a specified instrument drift rate is assumed between calibration intervals. If OLM is performed quarterly, as potentially required by NRC, the drift limits must be made more conservative so that the channel will be operable a quarter later considering the assumed drift rate. If the OLM is performed continuously, which may be approximated by daily checks, this additional allowance would not be required. This additional conservatism will not be implemented in the following example, but is mentioned for completeness.

The second restriction that affects the drift detection level is the model auto-sensitivity. Recall that the auto-sensitivity is a measure of how a sensor prediction responds to a drift of the respective sensor. If the auto-sensitivity is unity, the prediction is not robust to the drift, it will follow the drift, and the residual will be equal to zero even when a drift is present. Therefore, the extent to which an OLM system can detect a sensor fault is determined by two factors: (1) its predictive uncertainty and (2) its auto-sensitivity. This dependency will now be used in an example to fully develop the EULM detectability performance metric.

Suppose an empirical model of a sensor group has been developed, whose uncertainty of sensor i (95% CI) is found to be 1% of its nominal value. In the ideal situation, the sensor's prediction would be insensitive to input faults, and the smallest expected fault that the model could detect would be the magnitude of its uncertainty or 1%. In other words, when the residual grew to >1%, one could be 95% certain that it was due to a drift and not due to the uncertainty of the prediction. Because the ideal is rarely a reality, an empirical model's predictions can be expected to drift slightly when an input is faulted. In the case when the auto-sensitivity is greater than zero, the sensor must drift by more than 1% for the residual to have a magnitude of 1%. The equation for auto-sensitivity of the *i*th sensor is rewritten as the change in the prediction for a drifted input divided by the drift amount.

$$\Delta x_i^{drift} = x_i^{drift} - x_i \quad \textit{and} \quad \Delta \hat{x}_i^{drift} = \hat{x}_i^{drift} - \hat{x}_i$$

$$S_{A,i} = \left| \frac{\Delta \hat{x}_i^{drift}}{\Delta x_i^{drift}} \right| \qquad . \tag{2.6.1.1}$$

The residual is the difference between the input and the output with the input being the normal input plus the drift and the output being the normal output plus the portion of the drift (S_A) that appears on the prediction:

$$\text{Residual}_i = r_i = x_i + \Delta x_i^{drift} - \hat{x}_i - \Delta \hat{x}_i^{drift} \quad . \tag{2.6.1.2}$$

If the model is accurate, the prediction equals the actual value ($\hat{x}_i = x_i$). The contribution to the residual from the optimal prediction is usually due to noise and is often small compared to the propagation of the drift. In other words, the error of the predictions is small compared to the error that results when there is a drift in the input and the above equation simplifies to

$$r_i = \Delta x_i^{drift} - \Delta \hat{x}_i^{drift} \quad . \tag{2.6.1.3}$$

Using the equation for the auto-sensitivity, we have a measure of the residual in terms of sensitivity:

$$r_i = \Delta x_i^{drift} - S_{A,i} \left(\Delta x_i^{drift} \right) = \Delta x_i^{drift} \left(1 - S_{A,i} \right) \quad . \tag{2.6.1.4}$$

The above equation says that the residual is only equal to a percentage of the actual drift. For example, if the sensitivity is 0.2, then the sensor must drift by $\frac{1\%}{1-0.2} = 1.25\%$ for the residual to equal 1%. The EULM delectability equation can be written as

$$\boldsymbol{D}_i = \frac{\boldsymbol{U}_i}{\boldsymbol{E}_i[\boldsymbol{x}]} \cdot \frac{1}{1-\boldsymbol{S}_{A,i}} \ , \tag{2.6.1.5}$$

where

$\boldsymbol{U}_i$ is sensor *i*'s model uncertainty (95% CI),

$\boldsymbol{E}_i[\boldsymbol{x}]$ is sensor *i*'s expected or nominal value,

$S_{A,i}$ is sensor *i*'s auto-sensitivity.

The detectability has units of percentage and is a function of both model uncertainty and model auto-sensitivity. It could alternatively be in units of percent span if $\boldsymbol{E}_i(\boldsymbol{x})$ were replaced by the span of the *i*th instrument. If faults are being detected by examining the magnitude and uncertainty of the error, then the smallest detectable drift for a perfect predictor would be for an error of zero. Therefore, the detectability would be the uncertainty. Because there are sensitivity effects, the uncertainty must be augmented to get a more realistic estimate of the smallest detectable drift. The auto-sensitivity is used because it is related to the drift in a specific sensor. Typically it is also significantly larger than the cross-sensitivity metrics. In other words, when the detectability is corrected with the auto-sensitivity, it is corrected by a value greater than the cross-sensitivities. A greater auto-sensitivity ($\boldsymbol{S}_{A,i}$) causes the denominator of the above equation to get smaller and the detectability ($\boldsymbol{D}_i$) to get bigger; meaning that only larger drifts can be confidently detected. An optimal model would have an auto-sensitivity equal to zero.

2.6.2 Anomaly detection

Whereas the EULM method was used to determine the smallest sensor drift that can be identified, the anomaly detection performance metric is used to determine the smallest process parameter change that can be detected, such as changes caused by degradation or faults. Because we are not dealing with calibration verification, we will not directly evaluate the predictive uncertainty but will allow the algorithm to indirectly and iteratively determine it.

The goal in anomaly detection is to be able to detect subtle changes in process sensor expected values beyond those normally expected. The problem is how to determine whether the residual from a prediction is being caused by a faulted system or if it is due to normal process and instrumentation variations. Stated in a more general way, is the sequence of residuals being generated by a random process with a mean of zero or from a process with a nonzero mean due to some fault condition?

The Sequential Probability Ratio Test (SPRT) is a statistical technique developed by Wald [1947] that can be used to answer such a question. The objective of anomaly detection is to detect a system anomaly or failure as soon as possible with a very small probability of making a wrong decision. The SPRT procedure consists of testing whether a sensor is more likely to be in a normal mode H_0 or in a degraded mode H_1. The SPRT is optimal in the sense that a minimum number of samples are required to detect a fault (change in mean) existing in the signal.

The general procedure for the SPRT is to first calculate the likelihood ratio, which is given by the following equation where $\{\boldsymbol{x}_n\}$ is a sequence of consecutive $\boldsymbol{n}$ observations of $\boldsymbol{x}$.

$$\boldsymbol{L}_n = \frac{\text{probability of observing } \{\boldsymbol{x}_n\} \text{ given } \boldsymbol{H}_1 \text{ is true}}{\text{probability of observing } \{\boldsymbol{x}_n\} \text{ given } \boldsymbol{H}_0 \text{ is true}} = \frac{\boldsymbol{p}\left(\{\boldsymbol{x}_n\} / \boldsymbol{H}_1\right)}{\boldsymbol{p}\left(\{\boldsymbol{x}_n\} / \boldsymbol{H}_0\right)} \ . \tag{2.6.2.1}$$

The likelihood ratio is then compared to the lower ($\boldsymbol{A}$) and upper ($\boldsymbol{B}$) bound defined by the false alarm probability (α) and missed alarm probability (β) as follows.

$$\boldsymbol{A} = \frac{\beta}{1-\alpha} \quad \text{and} \quad \boldsymbol{B} = \frac{1-\beta}{\alpha} \ . \tag{2.6.2.2}$$

If the likelihood ratio is less than A then it is determined to belong to the system's normal mode $\boldsymbol{H}_0$. Conversely, if the likelihood ratio is greater than $\boldsymbol{B}$ it is determined to belong to the system's degraded mode $\boldsymbol{H}_1$ and a fault is registered. For this work, the SPRT is applied to the residuals between a sensor measurement and an empirical model's predictions of the parameter. The residuals are assumed to be normally distributed with a mean of 0 and variance of σ^2, which is an estimate of the random variation of the sensor signal. Therefore, the probability distribution function (pdf) for the normal mode of the residuals is given by:

$$\boldsymbol{p}\left(\boldsymbol{x} / \boldsymbol{H}_0\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{\boldsymbol{x}^2}{2\sigma^2}\right) \ . \tag{2.6.2.3}$$

From this description, two degradation modes are readily apparent and shown in Fig. 2-20. In the first plot there is a mean shift up (+***M***), and the second plot shows a mean shift down (–***M***). The random uncertainty is denoted by the spread of the Gaussian function. The SPRT simply determines if the residual sequence is more probably generated from the normal or faulted distributions.

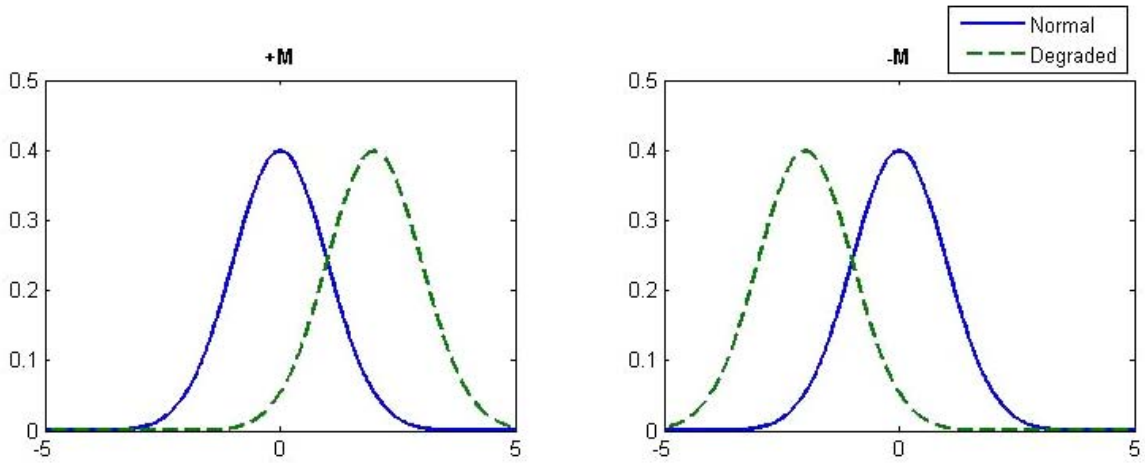


Fig. 2-20. Illustration of degraded modes for normal distribution.

The derivation of the likelihood ratios is beyond the scope of this paper but can be found in Humenik and Gross [1990]. The natural logarithm of the likelihood ratios are compared to ln(***A***) and ln(***B***) in most implementations of the SPRT algorithm and are listed in Table 2-1.

Table 2-1. Log likelihood ratio for a normal distribution with degraded mean

Degradation mode	Log likelihood ratio
+M	$\frac{\boldsymbol{M}}{\sigma^2}\left(\boldsymbol{x}-\frac{\boldsymbol{M}}{2}\right)$
–M	$\frac{\boldsymbol{M}}{\sigma^2}\left(-\boldsymbol{x}-\frac{\boldsymbol{M}}{2}\right)$

The magnitude of the sensor change caused by an anomaly or fault that can be reliably detected by the SPRT is defined as the magnitude of $\boldsymbol{M}$. If the observed values consistently lie near either $\pm\boldsymbol{M}$ the residual sequence is more likely to be generated from the $+\boldsymbol{M}$ distribution than from the normal distribution around 0.

The optimal $\boldsymbol{M}$ value is determined numerically by applying the SPRT to unfaulted, test data and locating the M value that results in a false-alarm probability that is nearest the theoretical false alarm probability α. If the optimal **M** value is estimated to be $\hat{\boldsymbol{M}}$, then the anomaly detection performance metric in percent is given by the following equation:

$$\boldsymbol{D}=\frac{\hat{\boldsymbol{M}}}{\boldsymbol{E}[\boldsymbol{x}]}\ . \tag{2.6.2.4}$$

Again, the detectability could alternatively be in units of percent span if $\boldsymbol{E}(\boldsymbol{x})$ were replaced by the instrument span.

2.7 Simulated Data Examples

This example presents an application of the three modeling techniques: AAKR, AAMSET, and AANN to a simulated data set. The model predictions and their performance measures will be calculated and compared.

For this example, t is defined as a set of 1000 uniformly spaced real numbers on the interval of [–0.5, 0.5]. The $\boldsymbol{t}$ value is used to define four true signals.

$$\boldsymbol{f}_1(\boldsymbol{t})=\sin(10\boldsymbol{t})\ . \tag{2.7.1}$$

$$\boldsymbol{f}_2(\boldsymbol{t})=\boldsymbol{t}\ . \tag{2.7.2}$$

$$\boldsymbol{f}_3(\boldsymbol{t})=\sin(5\boldsymbol{t})\ . \tag{2.7.3}$$

$$\boldsymbol{f}_4(\boldsymbol{t})=\boldsymbol{e}^{\sin(15\boldsymbol{t})}\ . \tag{2.7.4}$$

Finally, normally distributed random noise with a mean of 0 and variance of 0.4, $\boldsymbol{N_i}$ (0, 0.4), is added to each variable's true value to simulate process noise.

$$x_i = f_i(t) + N_i(0, 0.4) \qquad \text{with } i = \{1,2,3,4\} \ . \tag{2.7.5}$$

The simulated data were created by combining the variable's true values with independent realizations of their respective noise distributions, N_i (0, 0.4), and may be seen in Fig. 2-21.

The data set was randomly divided into training and test data sets. The training data were used to develop the empirical models, while the test data were used to evaluate their performance metrics.

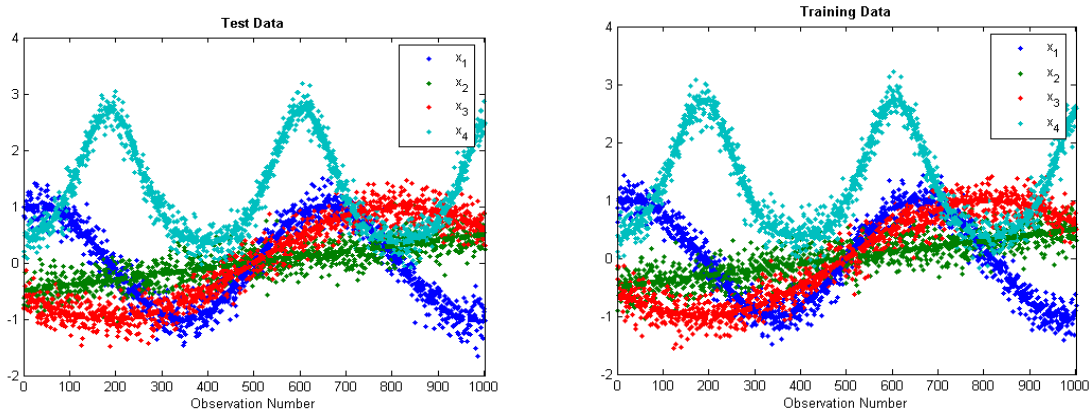


Fig. 2-21. Training and test data used to develop and compare model performance.

The correlation coefficient matrix presented in Table 2-2 shows that the second and third inputs are highly correlated with each other, while inputs 1 and 4 have only slight correlations with the other inputs. Note that process models with small correlation, such as these, would probably not result in a usable model for calibration monitoring. This simplified example was developed with functions that have no true dependencies and therefore have marginal performance. If more cycles of the sine wave were included, the performance would be worse yet. Higher correlated data would produce significantly better results.

Table 2-2. Correlation coefficient matrix

	x_1	x_2	x_3	x_4
x1	1	–0.232	0.099	0.241
x2	–0.232	1	0.922	–0.071
x3	0.099	0.921	1	–0.124
x4	0.241	–0.071	–0.124	1

For completeness the architectures of the three empirical models are as follows: (1) AAKR—500 memory vectors, bandwidth of 0.2, Euclidean distance function, and Gaussian kernel; (2) AAMSET—16 memory vectors and bandwidth of 1.0; and (3) AANN—4 mapping/demapping neurons and two bottleneck neurons. The predictions for x_3 the test data of each model are presented in Fig, 2-22.

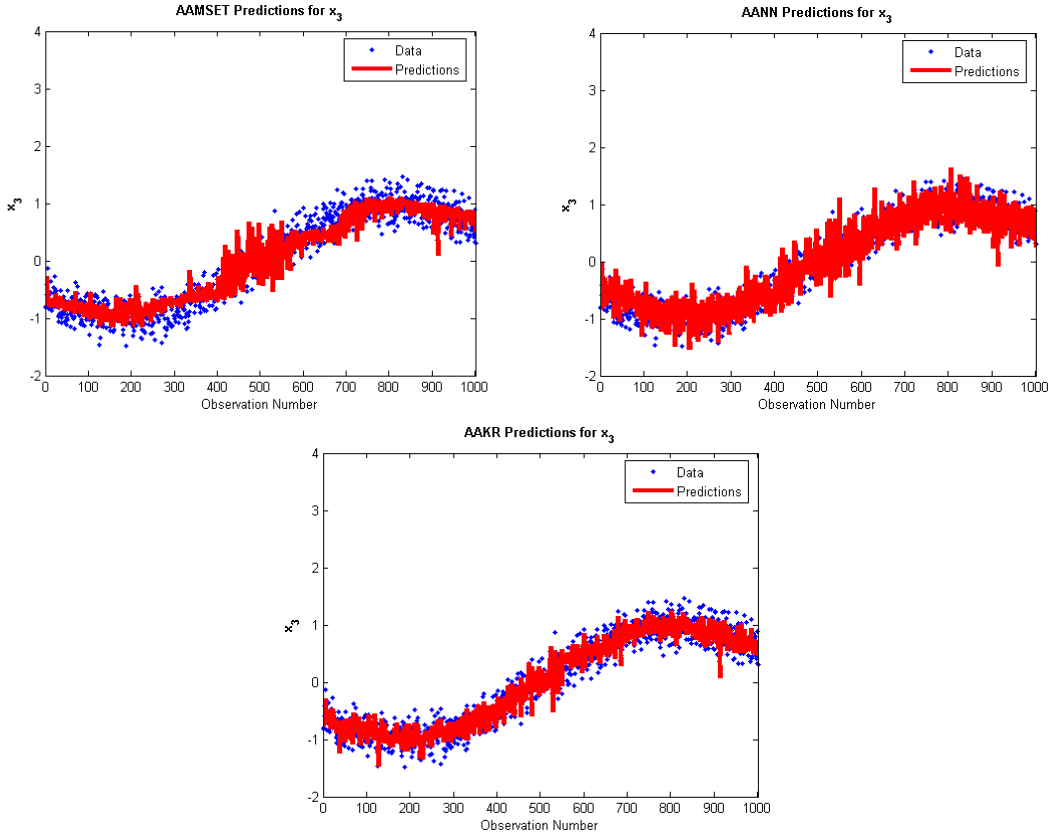


Fig. 2-22. AAKR, AAMSET, and AANN predictions for x_3.

Each of the variables was artificially, linearly drifted by 2 standard deviations such that their robustness and mean or 95% confident spill-over performance metrics could be calculated. The performance metrics for the three modeling techniques are displayed in Fig. 2-23 and listed in Table 2-3.

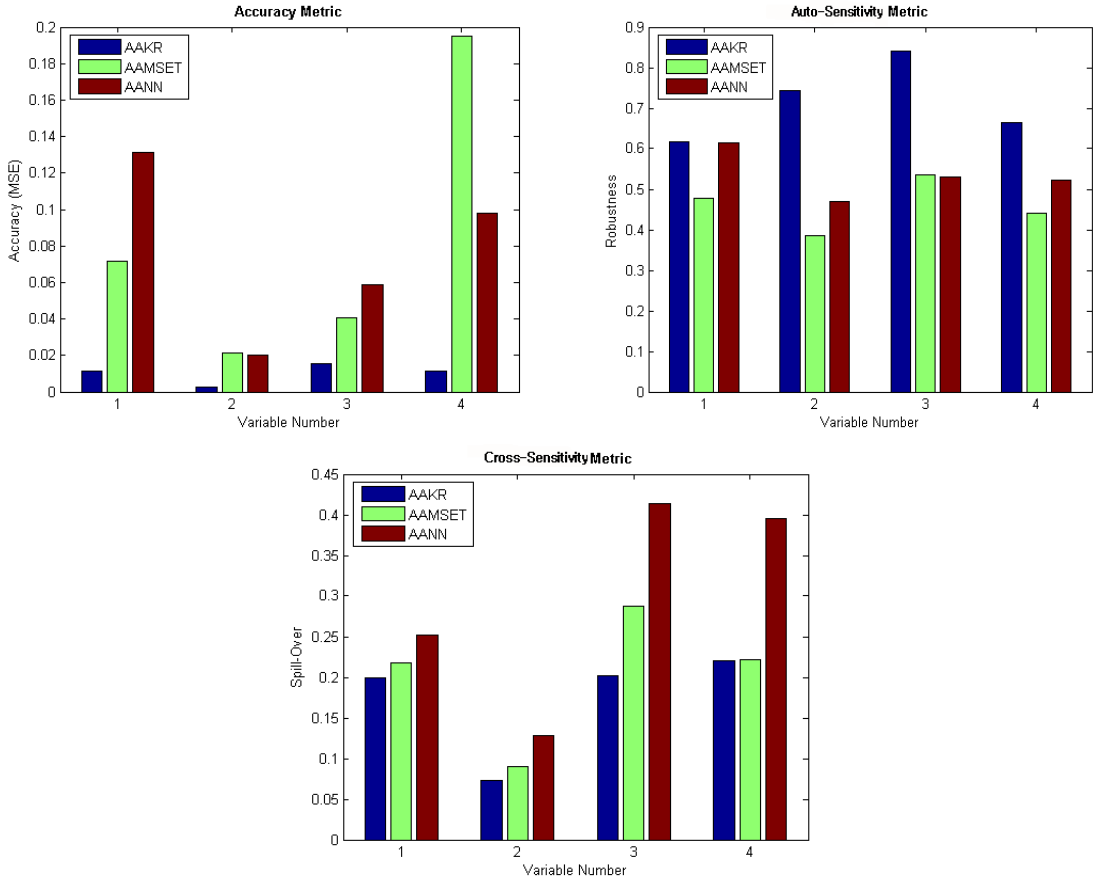


Fig. 2-23. Comparison of AAKR, AAMSET, and AANN performance metrics.

Table 2-3. AAKR, AAMSET, and AANN performance metrics

		x_1	x_2	x_3	x_3
AAKR	**Accuracy**	0.0114	0.0027	0.0153	0.0113
	Auto-sensitivity	0.616	0.744	0.8398	0.6647
	Cross-sensitivity	0.199	0.0729	0.2017	0.221
AAMSET	**Accuracy**	0.0717	0.0215	0.0409	0.1949
	Auto-sensitivity	0.4767	0.3851	0.5366	0.4408
	Cross-sensitivity	0.2176	0.0903	0.2871	0.2213
AANN	**Accuracy**	0.1311	0.0202	0.0585	0.0982
	Auto-sensitivity	0.6138	0.4692	0.529	0.5223
	Cross-sensitivity	0.2526	0.1277	0.4136	0.3957

It can be seen that the AAKR model has the smallest (best) accuracy metric, while the AAMSET and AANN models have the largest (worst) accuracy metric or worst accuracy performance.

Next, notice that the AAMSET model has the smallest auto-sensitivity metric, while the AAKR model has the largest (worst) auto-sensitivity metric. Again, the AANN performance is intermediate between AAKR and AAMSET. Therefore, for this example, the AAMSET model is the most robust, but sacrifices its accuracy performance. In other words, the predictions of the AAMSET model may be less accurate for good data, but less sensitive to changes in the inputs due to faults. It has a better ability to filter out faulty or corrupted signals.

Finally, notice that the cross-sensitivity performance of the models does not follow the pattern observed in the auto-sensitivity. Here, the AAKR model is best suited for estimating the true values of the other variables for a fault in one of the other variables. Also, notice that the AANN model has the worst cross-sensitivity performance, while the AAMSET model has an intermediate performance.

There is a well-known theory called the "No Free Lunch" theory (no-free-lunch.org) [Wolpert 1995]. It states that there is no model that is best suited for every data set. Some models perform better on certain data sets than others. Because of this, the performance on this one data set should not be taken to imply that a specific modeling technique is better than the others. From this one example, no conclusion can be made as to the expected performance on another data set. It is important to also realize that for a given model type the performance will also depend on model selection (e.g., bandwidth, training, and structure).

These performance characteristics are also illustrated in Fig. 2-24. For each model, two plots are displayed. The top plot is of the model's predictions of $\boldsymbol{x}_2$ for an artificial drift in $\boldsymbol{x}_2$. The lower plot is of the model's predictions of $\boldsymbol{x}_3$ for an artificial drift in $\boldsymbol{x}_2$, which shows how much a faulty sensor input has on the other sensor predictions. The top plot illustrates the model's auto-sensitivity (often referred to as) robustness performance, while the lower plot illustrates the model's cross-sensitivity (also known as spill-over) performance. Notice that as the drift progresses; the predictions for $\boldsymbol{x}_2$ of the AAKR model diverge from its un-faulted values more than that of the AAMSET model. This is an example of reduced robustness. Additionally, notice that the predictions for x_3 of the AAKR model are very near their modeled values, while the predictions of the AAMSET and AANN models diverge. This is called spillover: when a fault in one sensor spills over into another sensor's prediction.

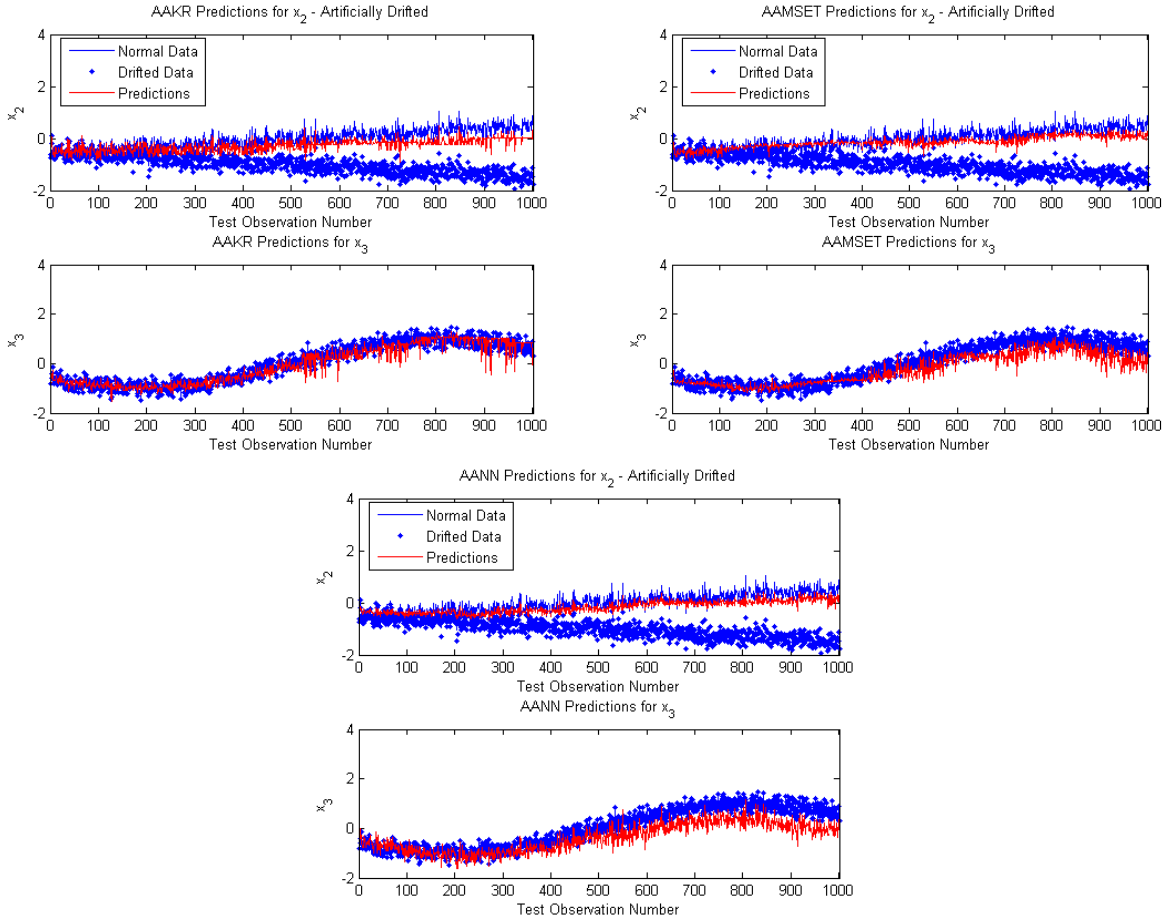
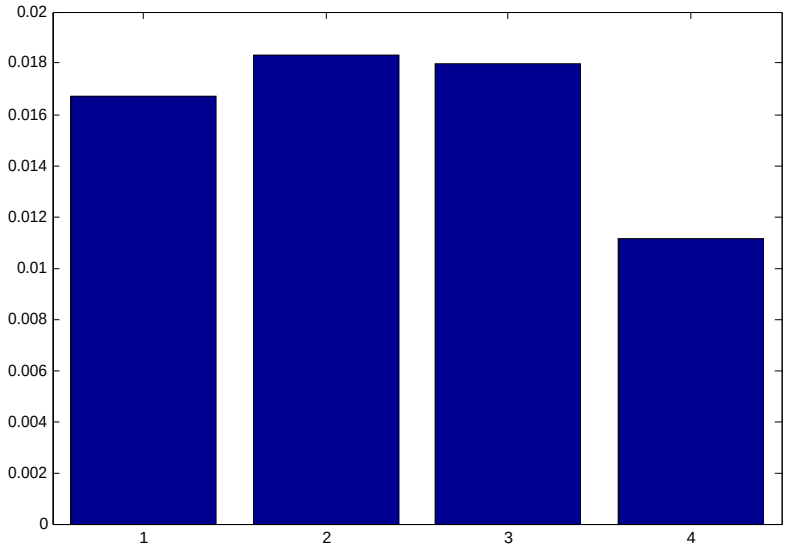
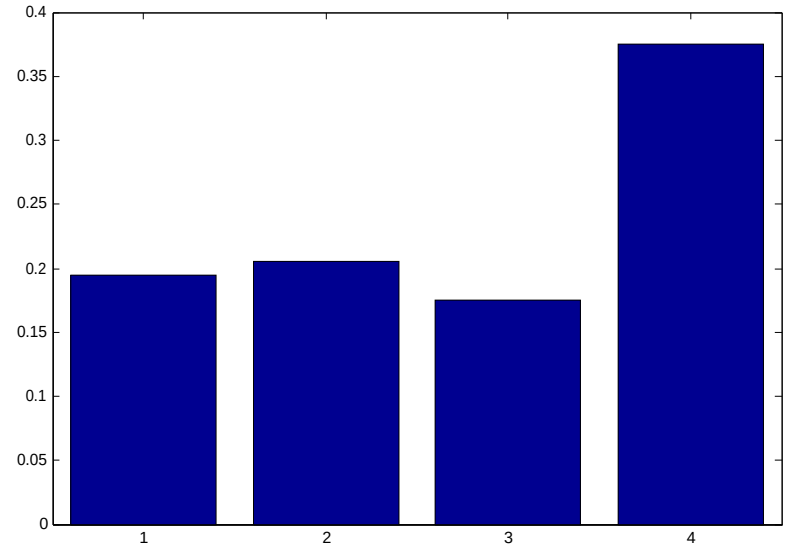


Fig. 2-24. Comparison of AAKR, AAMSET, and AANN performance with a drift in the second variable (x_2).

Finally, Fig. 2-25 displays the AAKR model's SPRT and EULM detectability metrics. These metrics represent an estimate of the smallest fault that can be detected according to the EULM and SPRT fault detection algorithms. In this example, the false alarm probability was set to 0.05 or 5%, and the missed alarm probability was set to 0.10 or 10% for the SPRT detectability.



(a)



(b)

Fig. 2-25. AAKR's (a) SPRT fault detectability and (b) EULM fault detectability for the simulated dataset.

We can see that the EULM fault detectability is almost an order of magnitude larger than the SPRT fault detectability. This difference can be attributed to the fact that the EULM fault detection algorithm has been designed to identify sensor calibration faults taking into consideration predictive uncertainties, while the SPRT does not consider uncertainties and is better suited in enunciating small process/system anomalies that may not be safety critical.

2.8 Model Summary

In this section the equations for the AAKR, AANN, and AAMSET models were derived. Along with presenting the theory of the auto-associative models, their application to OLM was discussed. In accordance with the "No Free Lunch Theorem" which explains why, over the set of all mathematically possible problems, each search algorithm will do on average as well as any other [Wolpert 1995], none of these models consistently outperform each other. Rather, the technique's performance depends on the specific input data being modeled. Still, all of these models have been found suitable for OLM and are being marketed in commercially available software. To compare the models, several metrics were presented to gauge the models' performances. However, one of the more, if not the most, important factors to determine a model's proficiency for OLM is its predictive uncertainty. The derivation of the fault detectability metrics showed the large role that a model's predictive uncertainty plays in determining its detectability metrics. If a model's uncertainty is too great, it must be excluded from use in an OLM system. Unfortunately, quantifying this uncertainty is an extremely complex task, and one of the more controversial issues surrounding OLM. Thus, Chaps. 3, 4, and 5 all focus on model uncertainty, the methods to evaluate it, and how it is incorporated into OLM detectability limits.

3. EMPIRICAL PREDICTIVE UNCERTAINTY

When empirical models are used to monitor safety-critical or high-value processes, model predictions should be accompanied by estimates of their uncertainty. This fact is required and strongly stated in the NRC's Safety Evaluation Report (SER) of OLM [NRC Project No. 669 2000]. Requirements 1, 3, 5, 6, and 7 of the SER pertain to the quantification of uncertainty associated with the plant parameter estimates. This section reviews each of these requirements and the remainder of the chapter presents methods for quantifying the uncertainty of empirical models used for OLM.

> **Requirement 1:**
> The submittal for implementation of the on-line monitoring technique must confirm that the impact of the deficiencies inherent in the on-line monitoring technique (inaccuracy in process parameter estimate single-point monitoring and untraceability of accuracy to standards) on plant safety be insignificant, and that all uncertainties associated with the process parameter estimate have been quantitatively bounded and accounted for either in the on-line monitoring acceptance criteria or in the applicable set point and uncertainty calculations.

Most aspects of Requirement 1 have been studied by the EPRI OLM user group and related institutions. The first volume of this NUREG/CR series discusses single-point monitoring in detail and cites the EPRI drift study [EPRI 2000]. Results from this study provided evidence that OLM was still valid even when the process is operating at a single point. However, to encompass the added uncertainty when the plant is operating with very little process change, a single-point monitoring penalty must be included with the other uncertainty terms in the drift allowance calculation. In the study, EPRI provides penalty values for many types and models of sensors. Furthermore, OLM looks only to extend the calibration interval and not to eliminate calibrations altogether. Because a minimum of one redundant sensor is calibrated at each fuel outage, traceability of accuracy to reference standards still exists. The predictive uncertainty calculation methods discussed in this report ensure that the process parameter uncertainty is quantitatively bounded when certain assumptions are met. Volume 3 of this series demonstrates how the methods degrade when the assumptions are not met. It is each individual plant's responsibility to make sure this uncertainty is accounted for when setting the OLM drift limits. Lastly, EPRI and others have only investigated the use of OLM in cases in which it will not affect set point calculations. The current strategy is to only apply OLM for sensor calibration extension when the predictive uncertainties are less that the drift limits used in the set point calculations.

> **Requirement 3:**
> The algorithm used for on-line monitoring be able to distinguish between the process variable drift (actual process going up or down) and the instrument drift and to compensate for uncertainties introduced by unstable processes, sensor locations, nonsimultaneous measurements, and noisy signals. If the implemented algorithm and/or its associated software cannot meet these requirements, administrative controls, including the guidelines in Section 3 of the topical report for avoiding a penalty for nonsimultaneous measurements could be implemented as an acceptable means to ensure that these requirements are met satisfactorily.

All of the algorithms currently being considered for OLM were designed with the intent that they distinguish between the process variable drift and the instrument drift. The three techniques described in this report (AANN, AAKR, and AAMSET) use the correlation of the instrument channels to differentiate the instrument drift from process changes. These techniques are not as susceptible to common mode drift because the correlation values for process drifts will be different than those for multiple instrument drifts. Furthermore, a model's sensitivity (its ability to make correct sensor predictions when a sensor is faulted) is computed and accounted for in EULM detectability algorithms.

Some signals may be too noisy or their predictions too uncertain for them to be candidates for OLM-based sensor calibration extension without changing set point calculations. Currently, EPRI and utilities investigating OLM indicate that normal manual calibrations would be used in these cases. Additionally, by applying an OLM model to historic data, sensors that are too noisy and fall outside the acceptable region can be identified before the OLM system is implemented. Any problems with data acquisition must be handled on a case-by-case basis.

Requirement 5:
Calculations for the acceptance criteria defining the proposed three zones of deviation ("acceptance," "needs calibration," and "inoperable") should be done in a manner consistent with the plant-specific safety-related instrumentation set point methodology so that using an on-line monitoring technique to monitor instrument performance and extend its calibration interval will not invalidate the set point calculation assumptions and the safety analysis assumptions. If new or different uncertainties should require the recalculation of instrument trip set points, it should be demonstrated that relevant safety analyses are unaffected. The licensee should have a documented methodology for calculating acceptance criteria that are compatible with the practice described in Regulatory Guide 1.105 and the methodology described acceptable industry standards for trip set point and uncertainty calculations.

The OLM allowance and uncertainties are to be applied n such a way that they do not affect the set point calculations. The uncertainties unique to OLM, such as the process parameter estimate uncertainty and single-point monitoring uncertainty, reduce the OLM drift allowance. EPRI Final Report 1007930 [2004] provides the methodology of the drift allowance calculations. This methodology ensures that the Technical Specification trip set point and related requirements do not require revision. Volume 1 of this series includes a discussion of OLM's relationship to set points.

Requirement 6:
For any algorithm used, the maximum acceptable value of deviation (MAVD) shall be such that accepting the deviation in the monitored value anywhere in the zone between parameter estimate (PE) and MAVD will provide high confidence (level of 95%/95%) that drift in the sensor-transmitter and/or any part of an instrument channel that is common to the instrument channel and the on-line monitoring loop is less than or equal to the value used in the set point calculations for that instrument.

Requirement 7:
The instrument shall meet all requirements of the above requirement 6 for the acceptable band or acceptable region.

The analytical uncertainty calculation methods presented in this report yield point-wise predictions, where each parameter estimate has an associated uncertainty estimate. However, when using Monte Carlo techniques the point-wise bias predictions form distributions that require quantification of 95%/95% levels. Engineering judgment must be used to determine the appropriate quantification method depending on the distribution. An example of one such method is to use the 95th percentile bias value, meaning that 95% of the bias values are smaller than the chosen metric. When combined with the variance estimate, it produces a conservative estimate of standard error of the predictive uncertainty. Subsequently, when the uncertainty magnitude is quantified, a t-statistic is chosen that provides 95% confidence level. This method ensures that Requirements 6 and 7 are met. Although there may be other appropriate methods for computing uncertainty besides those presented in this report, it is important that they compute the uncertainty components in a conservative manner (such as this 95th percentile method) if they are applied to OLM. Otherwise, the uncertainty may not meet a 95% confidence with 95% certainty and allow faulted sensors to stay in operation when they should be investigated further.

Overall, the SER requirements illustrate the importance of estimating the uncertainty of the plant parameter estimate so that the MAVD and ADVOLM can be established in a manner that ensures plant safety. (For a detailed explanation on calculating the MAVD and ADVOLM drift limits, the reader is referred to Volume 1 of this series). Overall, the quantification of this uncertainty is one of the most critical issues for the acceptance and safe-use of OLM. This section provides a general framework for estimating a model's predictive uncertainty. First, the theory of error propagation will be presented. Next, the problem of estimating uncertainty will be discussed in depth and will conclude in the production of a set of well-defined equations and requirements that need to be addressed in deployed OLM systems. This discussion will then be followed by two sections that address peripheral concerns in uncertainty analysis, specifically methods for estimating bias and denoising techniques used to estimate a set of data's true, noise-free values.

3.1 Problem Statement

Before the uncertainty of an empirical model may be estimated, the structure of uncertainty must be examined and several generalities addressed. This section will first define the problem of estimating uncertainty explicitly and then address several general problems that are readily apparent from this definition.

The purpose of the OLM system uncertainty analysis is to provide a traceable methodology for developing an estimate of the uncertainty for a given set of training data and model architecture.

Suppose that a sensor is monitoring a steady state value and contains normally distributed noise, or measurement error, with a mean and variance of 0 and σ_ε^2, respectively. This sensor's output will, therefore, have the probability distribution illustrated in Fig. 3-1, where $\boldsymbol{x_i}$ is the measured variable's "true" value.

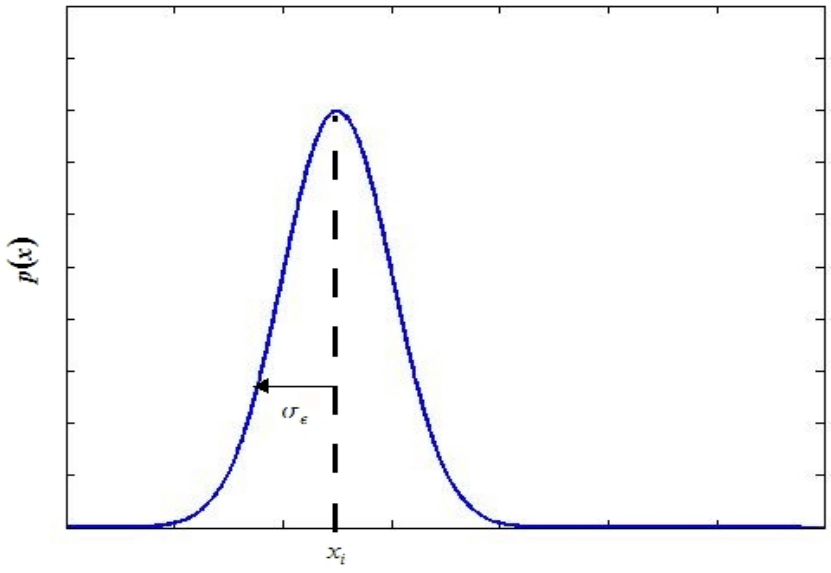


Fig. 3-1. Probability distribution for steady state sensor output.

If these sensor data were used to construct an empirical model, the distribution of the predictions will be slightly different than the sensor's distribution; because empirical modeling requires the trade-off of

generality and accuracy, the distribution of its predictions is inherently different than that of the data used to build the model. This difference may be seen in Fig, 3-2, where $\boldsymbol{E}(\hat{\boldsymbol{x}}_i)$ is the model's expected prediction and $\sigma_{\hat{x}_i}$ is the standard deviation (or square root of the variance) of the model predictions.

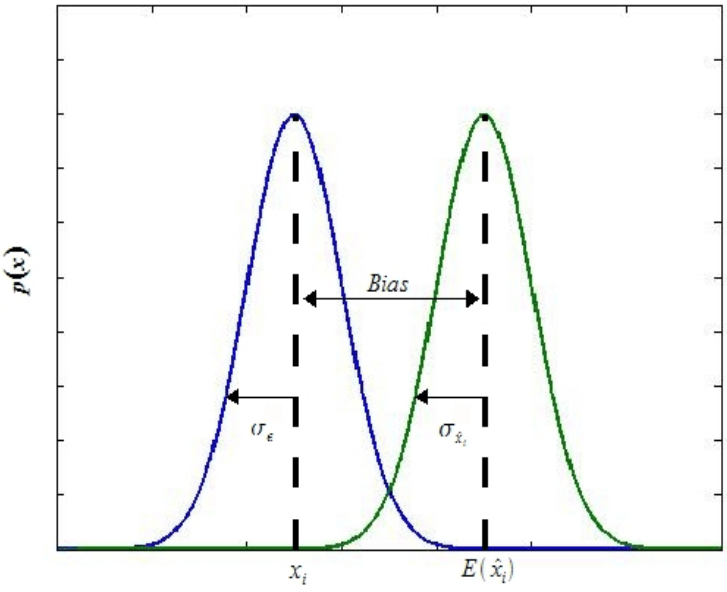


Fig. 3-2. Probability distributions for measured and predicted sensor values.

This difference can be analyzed through a bias-variance decomposition of the model's predictive uncertainty. In short, this decomposition states that the uncertainty of a model's predictions is a combination of its systematic error (bias) and random error (variance).

In the next section, the derivation of the bias-variance decomposition will be presented. Next, the uncertainty intervals that quantify the model uncertainty will be presented. Finally, a series of requirements for an uncertainty analysis in relation to OLM systems will be presented.

3.1.1 Bias-variance decomposition

The uncertainty of an empirical model's predictions may be broken into two major components: variance and bias. The variance measures the random error of the model's predictions, while the bias measures any systematic error. This decomposition will now be explicitly derived.

The derivation presented in this section is based upon that of Tamhane and Dunlop [2000]. To begin, the mean squared training error (MSE) can be calculated for different data sets. If it is calculated for the data used to construct the model, it is called the training MSE. If it is calculated for data not used in model development, it is called the test MSE. The training MSE is defined to be the average squared differences between a model's predictions and the target values (the measured values supplied to the model during development).

$$\boldsymbol{MSE}\left(\hat{\boldsymbol{x}}\right)=\frac{1}{\boldsymbol{n}}\sum_{i=1}^{\boldsymbol{n}}\left(\hat{\boldsymbol{x}}_i-\boldsymbol{x}_i\right)^2 \quad , \tag{3.1.1.1}$$

where

$\boldsymbol{x_i}$ is one of $\boldsymbol{n}$ target values;

$\hat{\boldsymbol{x}}_i$ is a model's estimate of $\boldsymbol{x_i}$.

The MSE measures how close a model's predictions are to their target values. The smaller the MSE, the closer the model fit is to the data. It is important to realize that when the MSE is calculated for the model with a data set that was used to construct the model ($\boldsymbol{MSE}_{\text{Train}}$), it may not be a good metric for the future predictive performance of the model. This is because a model may learn the random variations in the training data that are not duplicated in future data. This is called overfitting the training data: learning the noise and not just the underlying function. The future predictive performance of a model is quantified by the MSE calculated on data not used during training but indicative of future operations. The test MSE can be written in terms of the expected value:

$$\boldsymbol{MSE}_{\boldsymbol{Test}}=\boldsymbol{E}\left[\left(\hat{\mathbf{x}}-\boldsymbol{m}(\mathbf{x})\right)^2\right]+\varepsilon_{\boldsymbol{irr}} \quad . \tag{3.1.1.2}$$

$\boldsymbol{E}\left[\left(\hat{\mathbf{x}}-\boldsymbol{m}(\mathbf{x})\right)^2\right]$ is termed the reducible error, and $\varepsilon_{\boldsymbol{irr}}$ is called the irreducible error with

$$\boldsymbol{E}\left[\varepsilon_{\boldsymbol{irr}}\right]=0, \quad \boldsymbol{Var}\left(\varepsilon_{\boldsymbol{irr}}\right)=\sigma_\varepsilon^2 \quad . \tag{3.1.1.3}$$

The reducible error is the expected squared distance between the prediction $\hat{\boldsymbol{x}}$ and the true model of $\boldsymbol{x}$: $\boldsymbol{m}(\boldsymbol{x})$. The irreducible error is the difference between the true parameter value and the measured parameter value. It is caused by random process and measurement noise, and because it cannot be deterministically modeled, it is termed irreducible.

$$\sigma_\varepsilon^2=\boldsymbol{E}\left[\boldsymbol{x}_{\boldsymbol{true}}-\boldsymbol{x}_{\boldsymbol{measured}}\right] \quad . \tag{3.1.1.4}$$

The reducible error explains how adequately the model can represent the true function $\boldsymbol{m}(\boldsymbol{x})$. It depends only on the model architecture chosen, the training procedure, and the specific data set. The reducible error can be further decomposed into its bias and variance components.

$$
\begin{aligned}
\boldsymbol{E}\left[(\hat{\boldsymbol{x}}-\boldsymbol{m}(\boldsymbol{x}))^2\right] &= \boldsymbol{E}\left[\left(\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}]+\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\right)^2\right] \\
&= \boldsymbol{E}\left[\left(\{\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}]\}+\{\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\}\right)^2\right] \\
&= \boldsymbol{E}\left[\{\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}]\}^2+\{\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\}^2-2\{\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}]\}\cdot\{\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\}\right] \\
&= \boldsymbol{E}\left[\{\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}]\}^2\right]+\boldsymbol{E}\left[\{\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\}^2\right]-2\boldsymbol{E}\left[\{\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}]\}\cdot\{\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\}\right] \\
&= \boldsymbol{E}\left[(\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}])^2\right]+\boldsymbol{E}\left[(\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x}))^2\right] \\
&= \boldsymbol{E}\left[(\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}])^2\right]+(\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x}))^2 = \boldsymbol{Var}\ +\ \boldsymbol{Bias}^2
\end{aligned}
\quad . \quad (3.1.1.5)
$$

The simplification made in line four comes from:

$$
2\boldsymbol{E}\left[(\hat{\boldsymbol{x}}-\boldsymbol{E}(\hat{\boldsymbol{x}}))\square(\boldsymbol{E}(\hat{\boldsymbol{x}})-\boldsymbol{m}(\boldsymbol{x}))\right]=0\ , \quad (3.1.1.6)
$$

due to:

$$
\boldsymbol{E}\left[(\hat{\boldsymbol{x}})-\boldsymbol{E}(\hat{\boldsymbol{x}})\right]=\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{E}\left[\boldsymbol{E}[\hat{\boldsymbol{x}}]\right]=\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{E}[\hat{\boldsymbol{x}}]=0\ .
$$

This uses that fact that the mathematical expectation of a mathematical expectation is just a mathematical expectation [Heskes 1998]. For the same reasons,

$$
\boldsymbol{E}\left[\left(\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\right)^2\right]=\left(\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\right)^2\ . \quad (3.1.1.7)
$$

Because the variance is defined as the expected squared difference of a parameter from its expected value, the variance for a model's prediction may be written as

$$
\boldsymbol{Var}\left(\hat{\boldsymbol{x}}\right)=\boldsymbol{E}\left[\hat{\boldsymbol{x}}-\boldsymbol{E}[\hat{\boldsymbol{x}}]\right]^2\ . \quad (3.1.1.8)
$$

Next, recall that the bias is defined as the systematic error of a model. This may be written in equation form as the difference of a model's expected prediction from the true, target values:

$$
\boldsymbol{Bias}\left(\hat{\boldsymbol{x}}\right)=\boldsymbol{E}[\hat{\boldsymbol{x}}]-\boldsymbol{m}(\boldsymbol{x})\ . \quad (3.1.1.9)
$$

Equation (3.1.1.4) is the functional form of the bias-variance decomposition and serves as the foundation for the methodologies presented in this report. In the next section, this decomposition will be used to define the uncertainty intervals that quantify model uncertainty. However, the total uncertainty, $\boldsymbol{U}(\hat{\boldsymbol{x}})$, is a combination of model bias, variance, and the irreducible error:

$$\boldsymbol{U}(\hat{\mathbf{x}}) = \boldsymbol{Var}(\hat{\mathbf{x}}) + \left[\boldsymbol{Bias}(\hat{\mathbf{x}})\right]^2 + \sigma_\varepsilon^2 \ . \tag{3.1.1.10}$$

Recall that the irreducible error is the variance of the target about the true mean and cannot be controlled through the modeling process. It is caused by the random disturbances in the measurement process and is essentially the variance of the process and measurement noise. Process noise is the stochastic perturbation of a parameter about its true value that cannot be modeled deterministically. This type of noise is particularly evident in flow and level sensors, which have natural fluctuations due to "waves" about the intended value. Measurement noise refers to the noise inherent in the sensor. Electronic noise is the noise resulting from data transfer from the sensor to a central processor. These three noise contributors combine to cause random variances in the data. Often, the term is considered negligible and is neglected. However, to be sure this assumption holds, the amount of noise in the process must be quantified. This can be accomplished using one of the denoising techniques discussed later in this chapter.

3.1.2 Uncertainty intervals

Assume that a system is represented by the following equation:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \varepsilon \ . \tag{3.1.2.1}$$

where

- $\boldsymbol{y}$ is the measurable value for system variable being modeled;
- $\boldsymbol{x}$ are parameters used to infer the value of $\boldsymbol{y}$;
- $\boldsymbol{f}(\boldsymbol{x})$ is the true system value;
- ε is normally distributed random noise with a mean of 0 and variance σ_ε^2.

Although one desires the model to represent the true function $\boldsymbol{f(x)}$, only contaminated measurements y are available for model development.

The confidence interval (CI) is a quantification of the agreement of model predictions with the system's true value and is a measure of the magnitude of $\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}(\boldsymbol{x})$, where $\hat{\boldsymbol{f}}(\boldsymbol{x})$ is an estimate of $\boldsymbol{f}(\boldsymbol{x})$ for a given set of predictor variables $\boldsymbol{x}$. The confidence interval only accounts for the variance component of the uncertainty or the distribution of the quantity $\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}(\boldsymbol{x})$. For a confidence level of $1 - \alpha$ (e.g., $1 - \alpha = 95\%$), the confidence interval is defined by Tamhane and Dunlop [2000] to be the interval $[\boldsymbol{L}, \boldsymbol{U}]$, which includes $\boldsymbol{f}(\boldsymbol{x})$ with a preassigned probability of $1 - \alpha$:

$$\boldsymbol{P}\left[\boldsymbol{L} \le \boldsymbol{f}(\boldsymbol{x}) \le \boldsymbol{U}\right] = 1 - \alpha \ . \tag{3.1.2.2}$$

The prediction interval (PI) is a more practical measure of uncertainty and quantifies the agreement of model predictions with the actual measurements of the response (i.e., a measurement of the magnitude of $\hat{\boldsymbol{y}} - \boldsymbol{y}$, where $\hat{\boldsymbol{y}}$ is an estimate of $\boldsymbol{y}$). It can be seen that the confidence interval is enclosed within the prediction interval. In general, CI + ε = PI or more specifically $\left(\boldsymbol{f}(\boldsymbol{x}) - \hat{\boldsymbol{f}}(\boldsymbol{x})\right) + \varepsilon = \boldsymbol{y} - \hat{\boldsymbol{f}}(\boldsymbol{x})$. For an uncertainty level of $1 - \alpha$ the PI may be written as:

$$\boldsymbol{P}\left[\boldsymbol{L} \le \boldsymbol{y} \le \boldsymbol{U}\right] = 1 - \alpha \ . \tag{3.1.2.3}$$

An alternative definition of confidence and prediction intervals is that a CI is a measure of the uncertainty of a model's expected prediction, while the PI measures the uncertainty for each model prediction. In other words, because the expected response is $\boldsymbol{f}(\boldsymbol{x})$, specifically $\boldsymbol{E}(\boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{x})$, the CI is the uncertainty for the model's expected prediction. Conversely, since the PI uses the data itself, the prediction interval is the uncertainty for each individual model's prediction.

These definitions will now be expressed in terms of the quantities described in the previous sections, starting with the CI. In the following equations, it is assumed that methods exist for estimating the noise variance, the model prediction variance, and model bias.

For a confidence level of $1 - \alpha$, the CI may be written as:

$$\boldsymbol{E}\left(\hat{\boldsymbol{y}}\right) \pm \boldsymbol{t}_{\boldsymbol{n}-\boldsymbol{p},\alpha/2} \sqrt{\boldsymbol{Var}\left(\hat{\boldsymbol{y}}\right) + \boldsymbol{Bias}^2} \ , \tag{3.1.2.4}$$

where

$\boldsymbol{n}$	is the number of training observations;
$\boldsymbol{p}$	is the number of variables used to infer y;
$\boldsymbol{t}_{\boldsymbol{n}-\boldsymbol{p},\alpha/2}$	is the t-statistic that approximates the normal distribution for $\boldsymbol{n} - \boldsymbol{p}$ degrees freedom and confidence level $1 - \alpha$;
$\boldsymbol{E}\left(\hat{\boldsymbol{y}}\right)$	is the expected model prediction of $\boldsymbol{yy}$.

Notice that the CI does not contain a noise term (σ_ε^2). This illustrates the fact that CI measures the uncertainty in the model's expected prediction and does not account for the natural variability of the values being modeled.

For a confidence level of $1 - \alpha$, the PI may be written as:

$$\hat{\boldsymbol{y}} \pm \boldsymbol{t}_{\boldsymbol{n}-\boldsymbol{p},\alpha/2} \sqrt{\hat{\sigma}_\varepsilon^2 + \boldsymbol{Var}\left(\hat{\boldsymbol{y}}\right) + \boldsymbol{Bias}^2} \ , \tag{3.1.2.5}$$

where $\hat{\sigma}_\varepsilon^2$ is an estimate of the variance of the noise on $\boldsymbol{y}$.

The PI equation may be further simplified since for large numbers of training observations (Volume 3 of this NUREG/CR will explore how to determine the proper number of training observations) and a confidence level of 95%, the t-statistic approaches 2:

$$\hat{\boldsymbol{y}} \pm 2 \sqrt{\hat{\sigma}_\varepsilon^2 + \boldsymbol{Var}\left(\hat{\boldsymbol{y}}\right) + \boldsymbol{Bias}^2} \ . \tag{3.1.2.6}$$

Because the PI contains a noise variance term, it is by definition larger than the CI and is, therefore, a more conservative estimate of the model's uncertainty.

Now, it must be decided if the PI or CI is better suited for OLM uncertainty analyses. To make this determination, it must be known how the uncertainty estimates are used by the plant to monitor drift. Most

commonly, OLM monitors drift by determining when the residual exceeds a threshold, which is constructed by subtracting the model uncertainty from the drift limit. As explained in Chap. 1, the residual is the difference between the sensor and the model prediction. Oftentimes, noise in sensors measurements inflate the PI (which includes the noise variance term) to an amount that exceeds the threshold, even when no drift is present. Examples in Chap. 7 illustrate the large PIs produced when noisy sensors are included in the OLM model. Thus, it can be concluded that PIs are generally not directly applicable to OLM uncertainty quantification. In contrast, CIs usually give acceptable OLM uncertainties. The examples in Chap. 7 demonstrate the CI's suitability for OLM uncertainty analyses. However, there must be justification for using the CI. This needed justification comes from further examination of the CI and its application to the residual.

If a prediction's CI or PI includes the measured or true value respectively, it is said to "over" it. Therefore, a 95% PI implies that the *measured value* is covered 95% of the time; whereas a 95% confidence interval implies that the *true value* is covered 95% of the time. As previously discussed, the CI is desired for OLM uncertainty because its values typically fall below the measurement's drift limit. However, the CI assumes that the true value is known and cannot be applied if the assumption is not met. If the system is sampled and a denoising technique applied, a good estimate of the true value can be obtained. If filtering is used, the bandwidth must be selected so that the noise (σ_ε^2) is removed, but the process dynamics are retained. To do this, the system must be sampled faster than the process dynamics and the filter set appropriately. However, the residual can be filtered without these requirements. The residual can be filtered because the process dynamics are in both the variable and estimate and have been canceled out when forming the residual. The only dynamics in the residual are the drift dynamics. The residual contains the drift and the random noise (σ_ε^2) that cannot be modeled. Because drift dynamics are very slow, a lower sample rate is required to filter the residual than would be required to filter the variable. Thus, the residual can be filtered without losing the dynamics of interest. The filtered residual can be interpreted as the *true value* of the residual, because none of the dynamics have been lost and only the noise removed through filtering. This fact means that a CI can be appropriately applied to the filtered residual, because the true value is assumed to be known. By applying a CI to the filtered residuals, one can monitor for drift and the uncertainty is related to the prediction of drift and not the prediction of the variable's true value.

3.1.3 Requirements

A set of general requirements may be derived by the components of Eq. (3.2.2.6). It can be seen that three terms need to be estimated to develop a PI; specifically, the noise variance, the model prediction variance, and the model prediction bias. When calculating a CI, the noise variance calculation is not necessary.

The three contributing terms are redefined as follows:

1. σ_ε^2, the estimate of the noise variance on $\boldsymbol{y}$,
2. $\boldsymbol{Var}(\hat{\boldsymbol{y}})$ the variance of model predictions,
3. ***Bias***, the systematic error or bias of model predictions.

The first requirement of an OLM uncertainty analysis is to estimate the true process values. For this discussion, designate $\hat{\boldsymbol{f}}(\boldsymbol{x})$ is an estimate of $\boldsymbol{f}(\boldsymbol{x})$. This provides the required information needed to estimate the noise variance. First, an estimate of the noise distribution is calculated.

$$\hat{\varepsilon}_i = \boldsymbol{y}_i - \hat{\boldsymbol{f}}(\mathrm{x}_i) \quad \text{with} \quad \boldsymbol{i} = \{1,2,\ldots,\boldsymbol{n}\} \ . \tag{3.1.3.1}$$

Next, the noise variance is estimated by calculating the sample variance of the estimated noise distribution:

$$\hat{\sigma}_{\varepsilon}^2 = \frac{1}{\boldsymbol{n}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \left[\hat{\varepsilon}_{\boldsymbol{i}} - \boldsymbol{E}\left(\hat{\varepsilon}_{\boldsymbol{i}} \right) \right]^2 \quad . \tag{3.1.3.2}$$

Equation (3.1.3.2) reduces to

$$\hat{\sigma}_{\varepsilon}^2 = \frac{1}{\boldsymbol{n}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \left[\hat{\varepsilon}_{\boldsymbol{i}} \right]^2 \quad , \tag{3.1.3.3}$$

because, as explained in the previous section, $\boldsymbol{E}\left(\hat{\varepsilon}_{\boldsymbol{i}} \right) = 0$.

The variance of the model predictions may be found by either applying the analytic equations or using a Monte Carlo resampling technique. Each of the estimation methodologies has its own requirements, which will be discussed in Chaps. 5 and 6, respectively.

The third and final contributing factor to the PI is the systematic error of the model predictions. Two methodologies may be used to estimate this quantity. The details of these methodologies will be discussed in Sect. 3.2.

The result of this discussion is the identification of three values that must be estimated to develop empirical model prediction intervals:

1. an estimate of the "true" process data must be obtainable (see Sect. 3.3),
2. the model prediction variance must be estimated (see Sects. 4 and 5), and
3. the model bias must be estimated (see Sect. 3.2).

3.2 Methods for Estimating Bias

This section considers two empirical methods for estimating empirical model bias. It is important to note the methods presented in this report are not the only approaches to bias estimation. NIST Technical Note 1297 [1994] provides a thorough reference of some of the many other ways bias can be calculated. The methods in this report are included because they are the most frequently used methods by researchers in this field. The first method makes use of the statistical definition of bias, while the second uses the bias-variance decomposition equations derived in Sect. 3.1.1.

3.2.1 Direct bias calculation

The first method of bias estimation makes use of the following definition:

$$\boldsymbol{Bias} = \boldsymbol{E}\left(\hat{\boldsymbol{x}} \right) - \boldsymbol{x}_{\boldsymbol{true}} \quad . \tag{3.2.1.1}$$

Such a method is called direct bias calculation. It is shown schematically in Fig. 3-3.

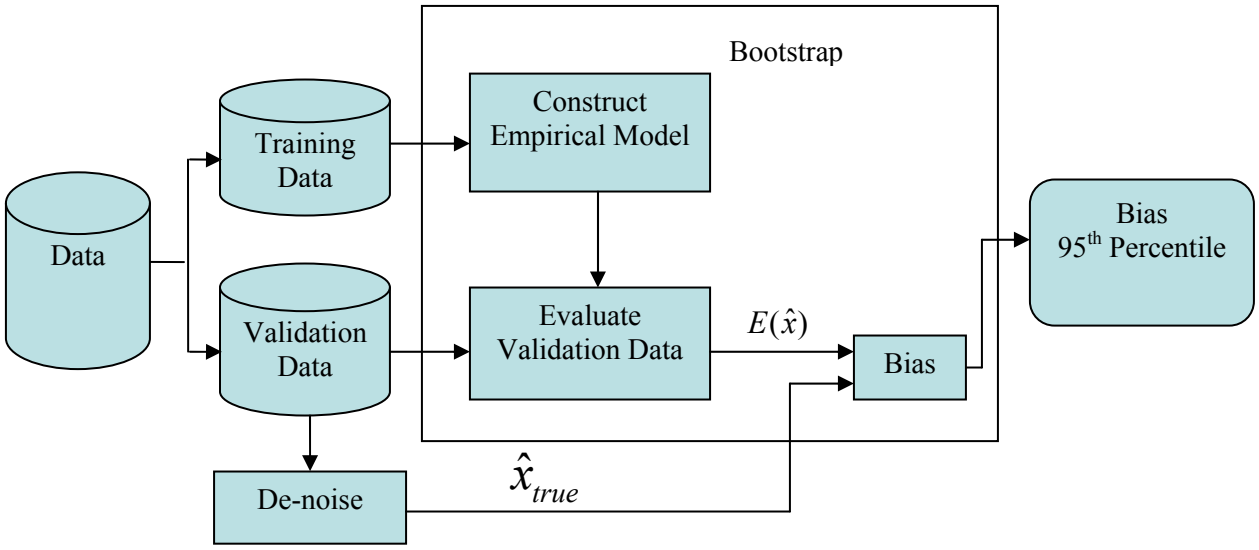


Fig. 3-3. Schematic diagram of the direct bias estimation.

A key idea behind this bias estimation method is that the sensor measurements are filtered to obtain an estimate of the virtually noise-free data and the parameters of the noise distribution. Using the obtained "noise-free" measurements and the estimated variance of the noise, bootstrapping, or a similar Monte Carlo technique, is performed, and the training data are used to construct multiple models. (Note that the bootstrap and other Monte Carlo simulation methods are discussed more fully in Chap. 5.) Then, these models are applied to the validation data to obtain new model predictions. Assuming the "noise-free" filtered data to be the true data, the difference between the "noise-free" observations and the model predictions is computed at a given point of interest for each of the many models. This calculation yields an estimate of the bias for each of the bootstrap models. This distribution of bias estimates could simply be averaged to obtain a single value for the bias. However, it is more conservative to use the top 95th percentile value of the bias estimates when calculating the single bias value. If the 95th percentile largest bias is used, then the interval will cover 95% of the actual errors. In comparison to a simple average, this 95th percentile calculation method produces a more appropriate and conservative estimate of the bias, which is desired for OLM. This "worst-case" estimate of the bias ensures that SER Requirements 6 and 7 [which state that both the ADVOLM and MAVD provide high confidence (level of 95%/95%) that drift in the sensor-transmitter is less than or equal to the value used in the set point calculations for that instrument] are met. Provided that n estimates have been made of x, this bias estimation method is given by:

$$\boldsymbol{Bias}(\boldsymbol{x}) \approx \text{Upper 95th percentile of} \left[\sum_{i=1}^{n} \hat{\boldsymbol{x}}_i - \boldsymbol{x}_{denoised} \right] . \qquad (3.2.1.2)$$

The denoised data are assumed to be approximately equal to the true values of the measured quantity. Finally, the estimate of the model bias is obtained as the difference between the denoised data and the model prediction.

To illustrate this bias estimation method, an example based upon synthetic data for three nonredundant sensors will now be presented. The following equations are taken to be the true system function that produces the true noiseless data that, in reality, are always unknown to the experimenter.

$$y_1(t) = 2 - 2t + 2\exp\left(\frac{(t-.5)^2}{.095^2}\right) \tag{3.2.1.3}$$

$$y_2(t) = 2 - 1.5t + 2\exp\left(\frac{(t-.5)^2}{.085^2}\right) \tag{3.2.1.4}$$

$$y_3(t) = 2.1 - 2.5t + 2\exp\left(\frac{(t-.5)^2}{.06^2}\right). \tag{3.2.1.5}$$

As can be seen in Fig, 3-4, the system functions produce three data sources of highly correlated quantities. The simulated signals feature highly linear regions, as well as regions of relatively large curvature (a Gaussian peak). Because both linear and nonlinear behaviors have been incorporated into this example, it is expected to reveal all aspects of bias estimation with the direct estimation method.

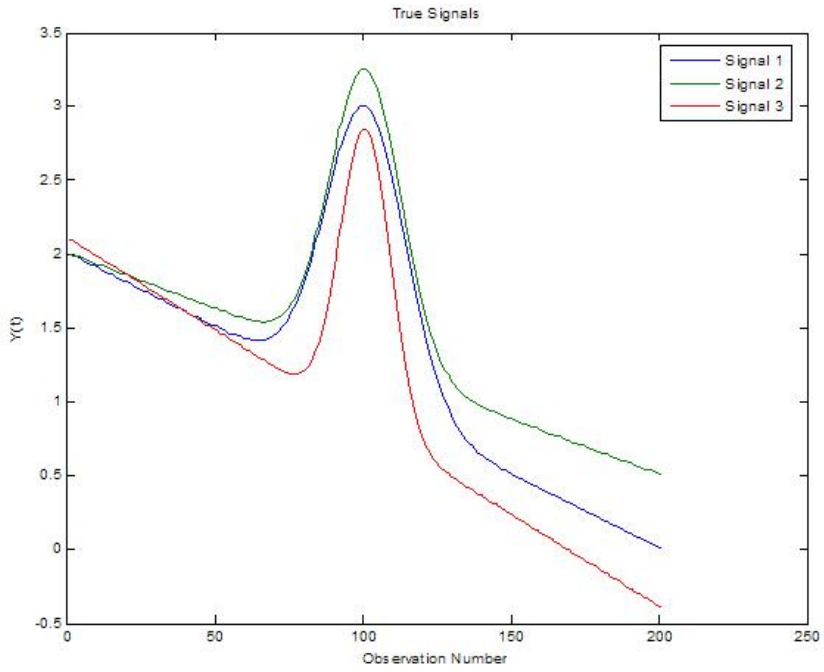


Fig. 3-4. True noiseless data.

To simulate real-world measurements, the true models are contaminated by Gaussian noise with a constant variance.

$$\tilde{y}_1(t) = y_1(t) + \varepsilon_1 \tag{3.2.1.6}$$

$$\tilde{y}_2(t) = y_2(t) + \varepsilon_2 \tag{3.2.1.7}$$

$$\tilde{y}_3(t) = y3(t) + \varepsilon_3 \ . \tag{3.2.1.8}$$

Here, ε_I are independent and identically distributed (i.i.d.) Gaussian random variables with a mean of 0 and a variance of $\sigma_\varepsilon^2 = 0.04$. Gaussian noise is used because the noise would be due to several sources and the Central Limit Theorem states that the resulting distribution tends toward Gaussian as the number of sources tends toward infinity. Also, note that $i = \{1, 2, 3\}$. The results of this addition may be seen in Fig. 3-5.

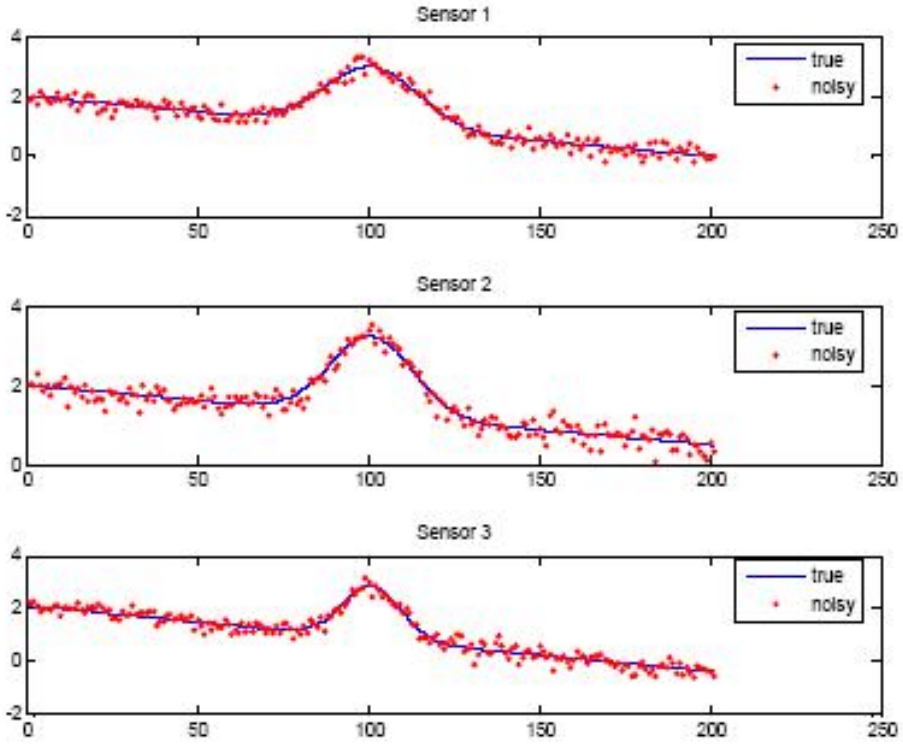


Fig. 3-5. Three sources of simulated noisy data.

The major task in the direct estimation technique is the denoising procedure, because the most important issue upon which the final estimated bias depends is how close the denoised estimates are to the true values. Ideally, the denoising provides the exact true values of the measured quantity so that the difference between the model predictions and the denoised data in the final result is only due to uncertainty inherent in Monte Carlo simulations.

A number of methods have been developed to filter random noise out of contaminated data, the simplest of which are median filtering and local averaging. Section 3.3 of this document considers two of the more sophisticated methods: wavelet denoising and Independent Component Analysis (ICA). These techniques are discussed and compared in the next section. Additional denoising investigations are

included in Volume 3 of this series. In that study it is shown that if the denoising technique is applied in an appropriate manner, the choice of technique has little impact on the uncertainty prediction.

An alternative denoising method is used in this numerical example: local polynomial smoothing. One major assumption of this smoothing technique is that the true generating model is assumed to be smooth and have a smooth first-order derivative. A detailed discussion of local polynomial smoothing can be found in Wand [1994]. In short, local polynomial smoothing approximates the target function by fitting a certain order polynomial in the vicinity of a given query point. It is simply a weighted average with the weighting determined by how "close" a point is to the point in question. A widely known local polynomial smoothing method is the Nadaraya-Watson kernel estimator [Nadaraya 1964, Watson 1964], as follows:

$$\hat{\boldsymbol{x}}(\boldsymbol{t}) = \frac{\sum_{i=1}^{n} \boldsymbol{K}_h(\boldsymbol{t}-\boldsymbol{t}_i)\,\boldsymbol{x}_i}{\sum_{i=1}^{n} \boldsymbol{K}_h(\boldsymbol{t}-\boldsymbol{t}_i)} = \sum_{i=1}^{n} (\boldsymbol{w}_i, \boldsymbol{x}_i) \ , \tag{3.2.1.9}$$

where n is the number of observations of $\boldsymbol{t}_i$,

$\boldsymbol{t}$ is the query point,

$\boldsymbol{K}_h(\boldsymbol{u})$ is a kernel function possessing the following properties:

$$\int \boldsymbol{K}_h(\boldsymbol{u})\,d\boldsymbol{u} = 1 \quad \text{and} \quad \int \boldsymbol{u}\boldsymbol{K}_h(\boldsymbol{u})\,d\boldsymbol{u} = 0 \ .$$

The most commonly used kernel function is the Gaussian function, which is given by the following equation:

$$\boldsymbol{K}_h(\boldsymbol{u}) = \frac{1}{\boldsymbol{h}\sqrt{2\pi}} \exp\left(\frac{\boldsymbol{u}^2}{2\boldsymbol{h}^2}\right) \ . \tag{3.2.1.10}$$

The bandwidth parameter, $\boldsymbol{h}$, controls what is considered "close" and therefore, also determines the smoothness of the estimates. The most common choice for the similarity argument $\boldsymbol{u}$ is the Euclidean distance between the query vector and the given observations.

This numerical example makes use of the local linear smoothing to obtain noise-free values of the simulated measurements. The optimal bandwidth parameter is obtained through minimizing the Leave-One-Out Cross-Validation metric [Stone 1974, Chapelle 2002]. The results of the denoising procedure are shown in Fig. 3-6.

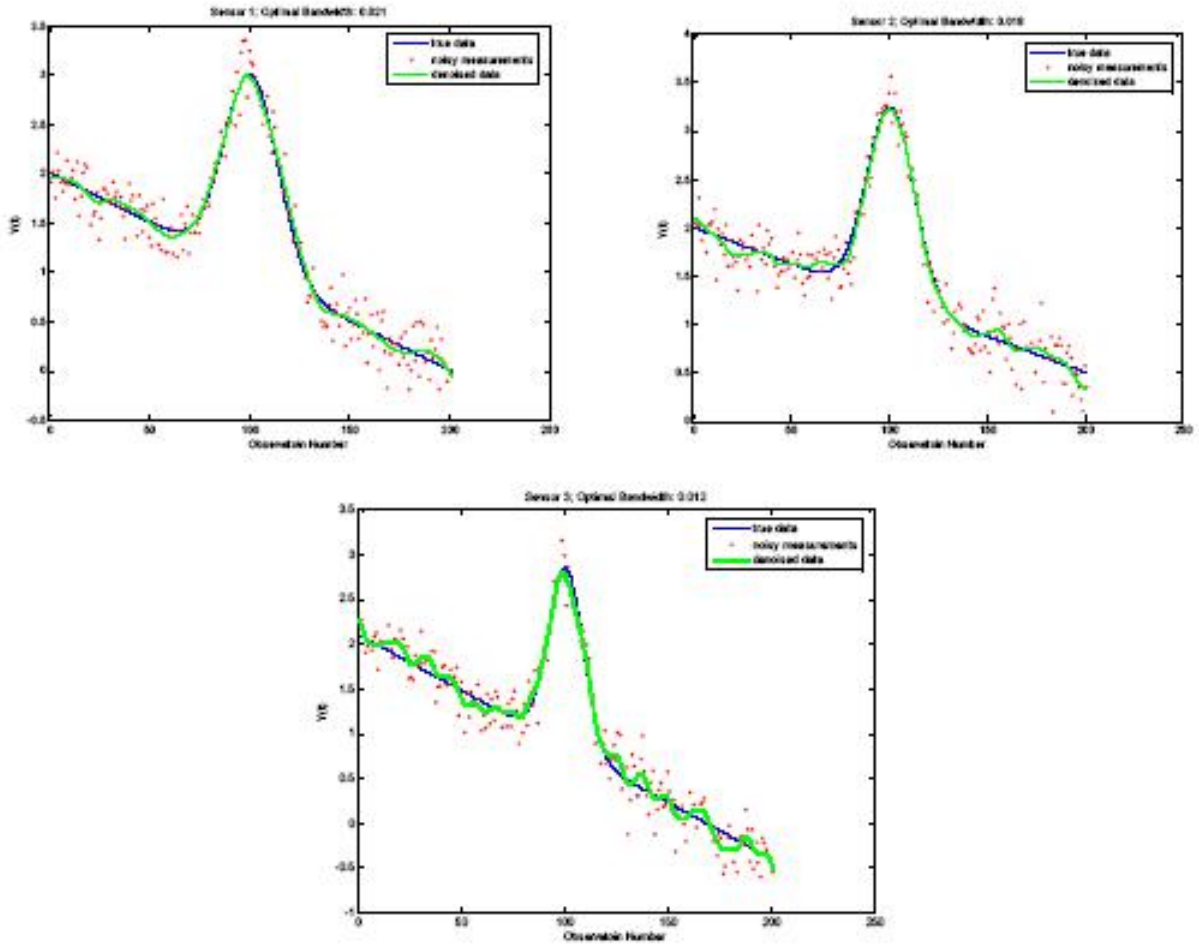


Fig. 3-6. Results of the local linear smoothing.

A brief overview of Monte Carlo techniques is now given with a more complete description given in Chap. 5. Monte Carlo techniques can be used to estimate the variance portion of the uncertainty. A good reference for these techniques is Efron and Tibshirani [1993]. The basic idea is to simulate the uncertainty of the empirical model by constructing a large number of example empirical models and quantifying the variance of their predictions. Each model is constructed by either using a random sample of the training data (bootstrap), or by modeling the deterministic and random portions of each signal and creating simulated samples. The deterministic portion is estimated through denoising the measured signals, and the random portion is estimated by modeling the noise with a distribution. Simulated samples are constructed by combining the denoised signal with a sample from the modeled distribution.

In this example, the smoothed data are taken to be the "noise-free" data. The estimate of the noise is obtained by subtracting the smoothed data from their noisy counterparts resulting in an estimate of the sample noise. Then, the variance of the sample noise is calculated. Table 3-1 summarizes the estimated sample noise standard deviations. In this example the noise contaminating the observations is assumed to have a Gaussian distribution with zero mean. This assumption is important because the Monte Carlo sampling is performed by selecting random variables from the estimated distributed.

Table 3-1 Summary of noise standard deviation

Sensor No.	$\hat{\sigma}$	Estimation error $\left[\frac{\hat{\sigma}-\sigma_{true}}{\sigma_{true}}\times 100\%\right]$
1	0.186	7%
2	0.179	10%
3	0.165	17%

The Monte Carlo simulation is performed according to the following algorithm:

1. generate a set of normally distributed random values with zero mean and the desired standard deviations,
2. add the generated random values to the filtered data,
3. divide the noisy data into a training set and a validation set,
4. construct a data-based model using the training data,
5. use the model to produce corrected estimates of the validation query data,
6. estimate the bias between the validation data and query estimates, and
7. go to step 1 unless the desired number of Monte Carlo runs is reached.

In this example, the data-based model mentioned in step 4 is taken to be an auto-associative kernel regression-based estimator (AAKR) discussed earlier in Chap. 3. Because the AAKR model employs a kernel bandwidth parameter, an arbitrary value of 1 has been used in this numerical experiment. The parameter value of 1 is expected to be large enough to introduce a small bias into the model predictions. The Monte Carlo predictions are used to perform direct calculation of the model bias. Because truly noise-free data are known in this simulation, the true bias can be easily calculated as the difference between the true data and the model prediction. A single bias estimate is obtained by only considering the 95th percentile value of the calculated biases. Figure 3-7 shows the estimated bias and true bias. As can be seen, the estimates are relatively close to the true values, but deviate in certain regions. Signal biasing usually occurs in high-gradient areas, but it may also occur in areas where the training data do not adequately constrain the problem.

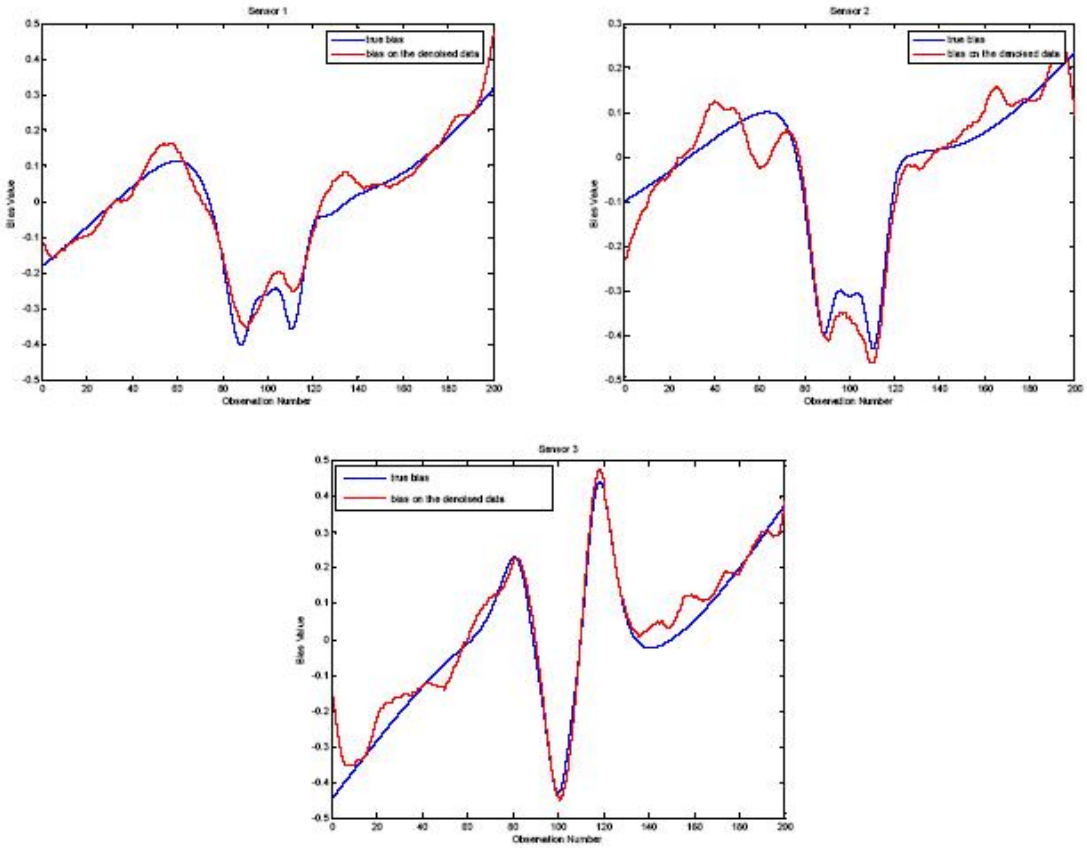


Fig. 3-7. Bias introduced by the AAKR model.

In this case, the deviations from the true bias are mostly caused by the imperfection of the smoothing procedure. The Monte Carlo simulation also allows the estimated bias to deviate from the true values since the Monte Carlo samples have not been performed on the true data but only the denoised values.

3.2.2 Application of bias-variance decomposition

The next example illustrates another method of bias estimation. This method is based upon the bias-variance decomposition equation [Eq. (3.2.2.1)] derived in Sect. 3.1.1:

$$\boldsymbol{U}\left(\hat{\boldsymbol{x}}\right)=\sigma_{\varepsilon}^{2}+\boldsymbol{Var}\left(\hat{\boldsymbol{x}}\right)+\left[\boldsymbol{Bias}\left(\hat{\boldsymbol{x}}\right)\right]^{2}\ . \tag{3.2.2.1}$$

Recall that the left-hand side of Eq. (3.2.2.1) is quantified by the MSE. Having a value for the MSE, one can obtain the estimate of the squared bias as follows:

$$\boldsymbol{Bias}\left(\hat{\boldsymbol{x}}\right)^{2}=\boldsymbol{MSE}\left(\hat{\boldsymbol{x}}\right)-\boldsymbol{Var}\left(\hat{\boldsymbol{x}}\right)-\sigma_{\varepsilon}^{2}\ . \tag{3.2.2.2}$$

The irreducible error term is present in Eq. (3.2.2.2) because the MSE is generally calculated using the validation data, rather than the training data. Because the validation data were not used to construct the

model, the ***MSE*** for the validation data is a measure of the model's true future predictive performance and requires that σ_ε^2 be included.

The model's prediction variance can be estimated using the Monte Carlo simulation, similar to that described in the previous example. The Monte Carlo simulation, in turn, can be used to evaluate the ***MSE***. The following algorithm is proposed to perform the ***MSE*** estimation:

1. denoise the data and estimate the irreducible error σ_ε^2,
2. divide the original noisy data into a training set and a validation set,
3. use Monte Carlo simulation to construct a data-based model with the training data,
4. obtain the estimates for the validation query data,
5. evaluate the squared difference between the noise-free signal and the model estimates,
6. go to Step 3 unless the desired number of Monte Carlo runs is reached,
7. calculate the variance of each validation sample over the Monte Carlo simulations,
8. compute an estimate for the bias for each run using Eq. (3.2.2.2), and
9. calculate the 95th percentile value of the bias estimates to obtain a single value for the bias.

The data-based model can be constructed with the data randomly sampled from the given measurements. The diagram depicted in Fig. 3-8 shows the bias estimation method based on evaluating the ***MSE*** of the validation data.

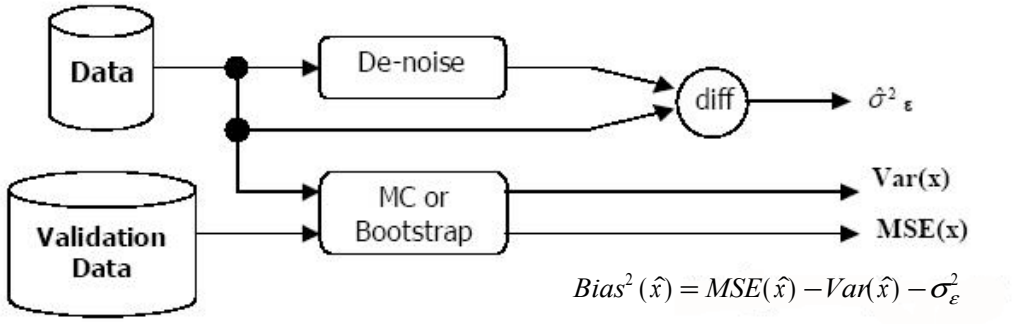


Fig. 3-8. Schematic diagram of the bias estimation through calculating the MSE.

The Monte Carlo simulation is the key algorithm of the bias estimation method. To illustrate this point, a series of numerical experiments will be performed. For the sake of clarity, the first numerical experiment is conducted using the truly noise-free data. Such a "toy" experiment should yield a clear understanding of the key ideas behind the estimation method. This example computes the bias of an AAKR model. As can be seen in Fig. 3-9, the estimated squared bias is very close its true value. This shows that the bias-variance decomposition equation may be used to estimate the squared bias of a model's predictions. The slight deviations of the estimates from the true bias values can be explained by the random nature of the Monte Carlo simulation involved in the estimation. There can also be seen to be a few relatively large departures from the true bias. They are likely to be due to the inaccurate Monte Carlo-based estimation of the prediction variance.

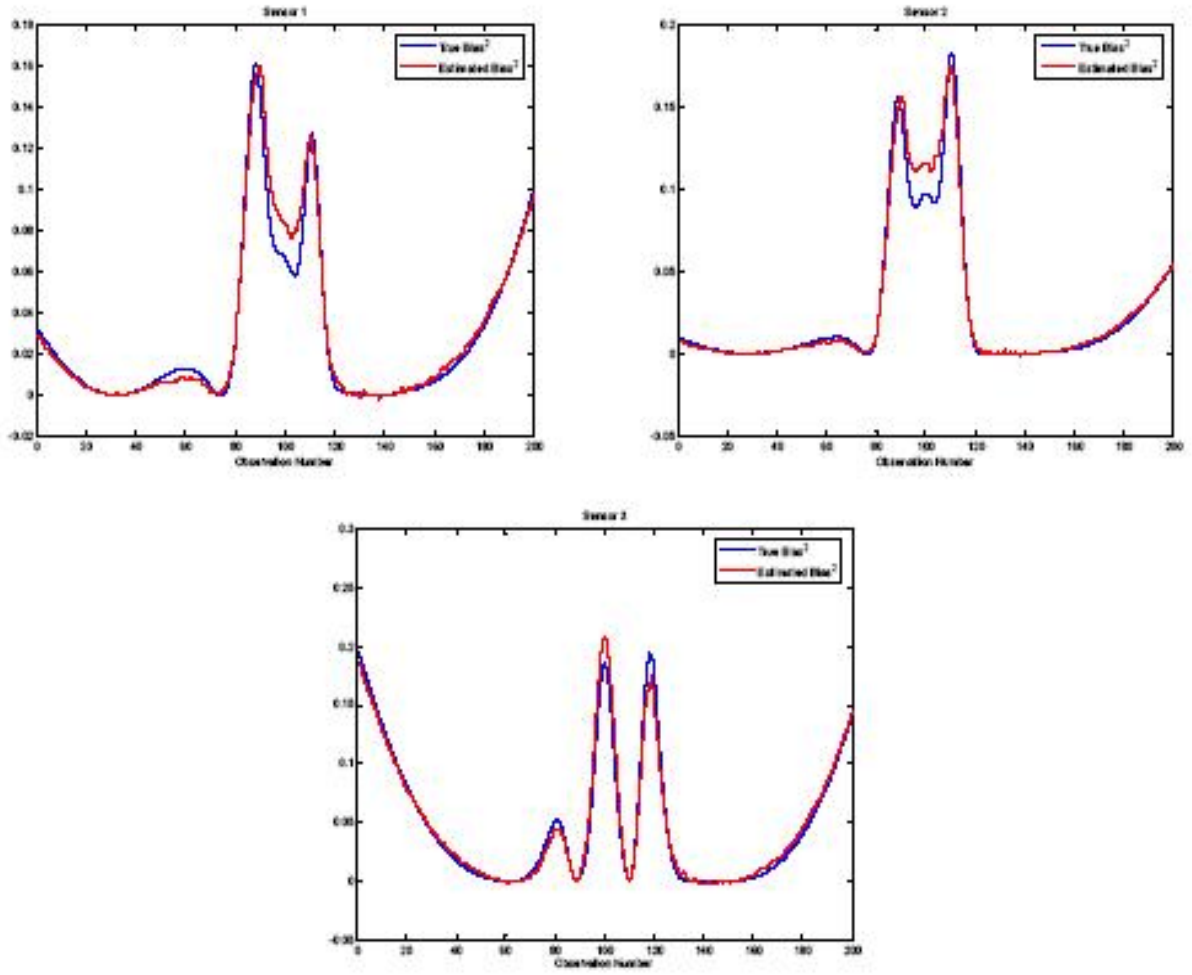


Fig. 3-9. Squared bias obtained with true noise-free data.

In the next experiment, the bias is calculated using unfiltered noisy measurements. Figure 3-10 shows the squared bias estimates. Because denoising was not performed, the bias is poorly estimated. This incorrect bias estimation would negatively impact the uncertainty calculation performed by an OLM system.

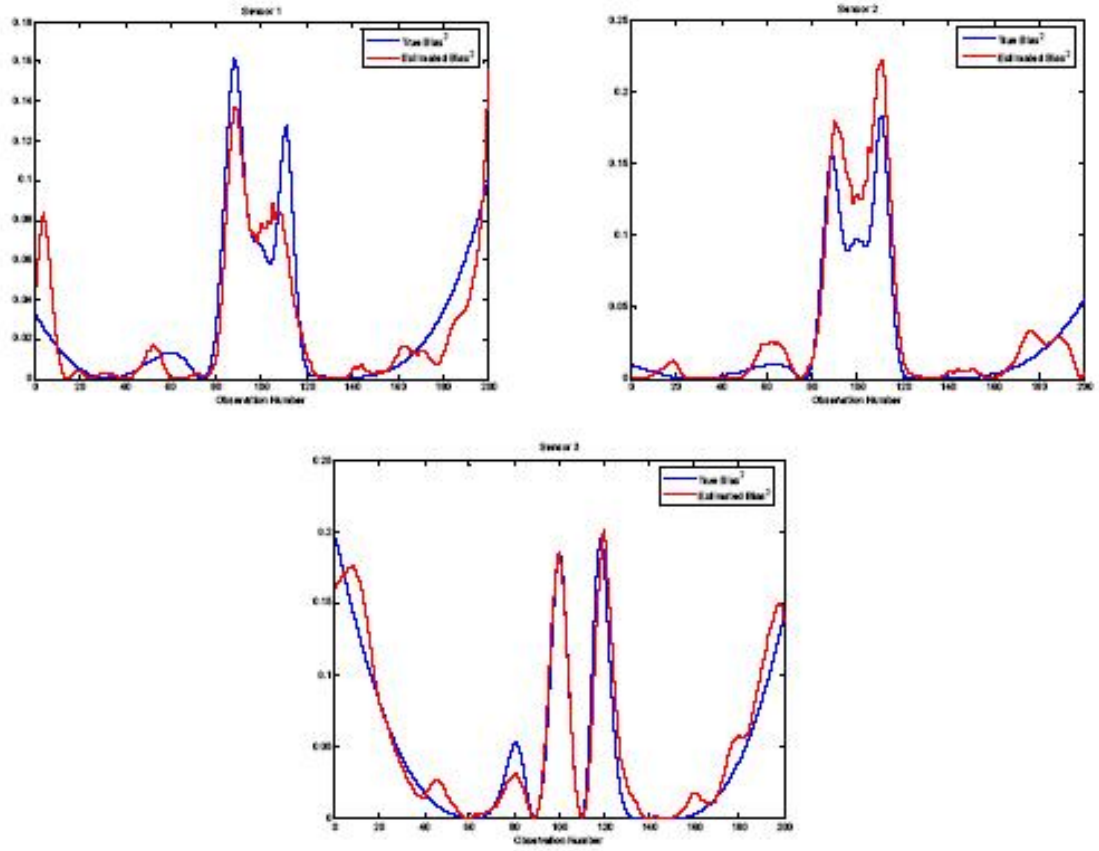


Fig. 3-10. Squared bias estimated on the observed data.

From the conducted experiments, it can be concluded that the accuracy of the considered estimation method depends on denoising the measurements. The closer the filtered data are to the true data, the more accurate the estimate of the model bias. These experiments also illustrated that the bias is largest in areas of high curvature. This is attributed to the fact that empirical models often have a tendency to oversmooth sharp peaks in the data. In nuclear power plants, the data are mostly steady state and smooth, so the biasing is usually smaller and more consistent.

3.3 Denoising Techniques

Denoising plays a critical role in an uncertainty analysis; however, Volume 3 of this series shows that the choice of denoising technique has limited effect on the prediction uncertainty when correctly applied. As described in Sect. 3.3, denoising is an important step in estimating the bias. It is also needed to estimate the value of σ_ε^2. This quantity is commonly termed the "irreducible error" because it cannot be controlled through the modeling process and is due to random disturbances in the measurement process. Essentially, σ_ε^2 quantifies the process noise. Noise is somewhat hard to define, and many individuals/organizations argue about its exact definition. In general, though, the "signal" is considered to be the information-carrying part of the input and "noise" to be any additional informationless part. In relation to nuclear power, noise is assumed to be normally distributed with a mean of 0 and variance σ_ε^2. Therefore, the only parameter necessary to quantify the noise is σ_ε^2. At first, estimating this quantity may

seem rather straightforward, but, unfortunately, many problems arise when using actual data to make this determination.

3.3.1 Filtering

Many previous analyses have simply employed a filtering technique to determine the noise level in a set of data. The filtered signal is subtracted from the unfiltered signal to isolate the noise. Although this method is often valid, several factors must be considered. The first is the sampling rate. If the data are not sampled at a rate faster than the process dynamics, then filtering would smooth the process dynamics and overestimate the noise level. To elaborate, consider Fig. 3-11, which presents temperature data from a nuclear power plant. In the figure, the units of time are not specified. If the time on the x-axis is in units of seconds or even minutes, then it would not be physically possible for the temperature to change quickly enough to create the large fluctuations exhibited by the data. In this case, filtering could be performed without losing information about the actual process. However, if the time is in days, or perhaps even hours, the process variable might vary at the rate shown in the data. Thus, filtering would not be allowed. Engineering judgment always must be used before applying a standard filtering technique to data because it requires prior knowledge about the data and the underlying process. Filtering should, therefore, only be used when there is certainty that the process dynamics will not be lost.

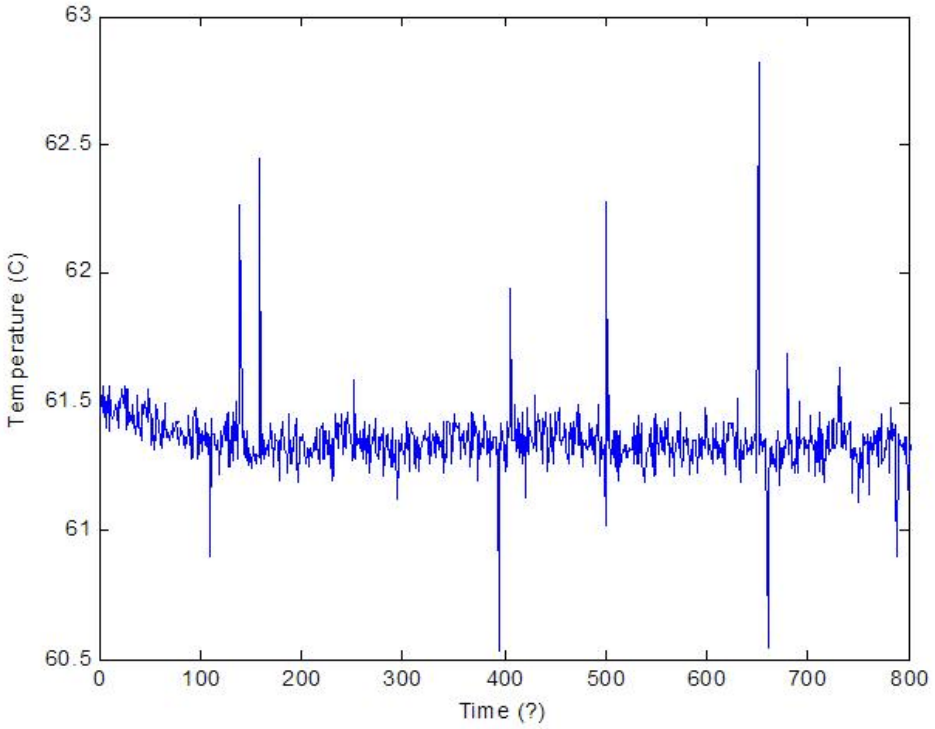


Fig. 3-11. Plot of temperature data.

Another potential problem with using standard filtering to estimate the noise level is that it cannot distinguish between common process noise and independent instrument noise. As depicted in Fig. 3-12, redundant sensor groups in nuclear plants contain common as well as independent noise sources. As shown in the figure, each redundant sensor contains a source signal indicative of the actual process (denoted by P in the figure). However, generally redundant sensors will not produce exactly identical readings, even though they are measuring the same process. The discrepancy between correctly operating

redundant sensor measurements is due to the noise contamination. Although given the term "noise," the common noise is actually reflective of the process. Some examples of common noise are water-level fluctuations due to turbulent flow or the high-frequency boiling component of a signal measured by a steam generator level detector. Each sensor also contains some unique independent noise. Independent noise is frequently caused by electrical noise contamination of the specific sensor signal. Because common process noise is indicative of a process variable's state, then the irreducible error should be comprised only of independent noise and not common process noise. The figure shows each sensor as having only a single common and independent noise source. In actuality, a sensor can be contaminated by multiple common and independent noise sources.

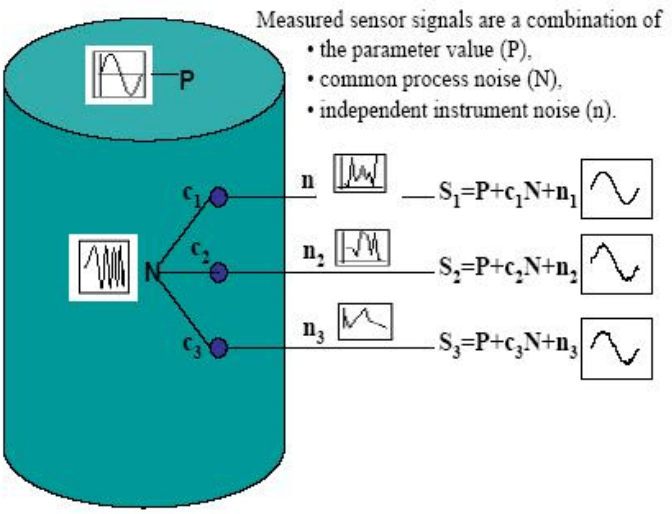


Fig. 3-12. Diagram of real world signals containing both common and independent noise.

When one is using conventional frequency-based filtering techniques, there is no guarantee that only independent noise is removed. Frequently, common noise, which should not be filtered, is also removed because both may contain similar frequency content. Because there is usually little prior knowledge that is useful in differentiating between common and independent noise, it is difficult to select proper window sizes or filtering parameters. Generally, the small addition of common process noise in the error term is considered negligible. However, in situations where the common process noise is exceedingly large, the predictive uncertainty may be inflated to the point where the OLM drift limits are exceeded. Therefore, standard smoothing methods may be unsuited for OLM applications. Two methods that may be better suited for this application are wavelet denoising and independent component analysis (ICA).

3.3.2 Wavelet denoising

The wavelet denoising technique investigated in this work was developed by Miron [2001] and was initially used to produce artificial data sets that closely resembled raw nuclear plant data. This approach decomposes a signal, $\boldsymbol{S}$, into an approximation, $\boldsymbol{A}$, and set of detail components, $\boldsymbol{D}$. If the decomposition level is $\boldsymbol{L}$, then this approximation may be written as:

$$\boldsymbol{S}_L = \boldsymbol{A}_L + \sum_{i=1}^{L} \boldsymbol{D}_i \ . \qquad (3.3.2.1)$$

The approximation component represents the low-frequency (system behavior) portion of the data, while the detail components represent the high-frequency (noise) portions of the decomposed approximations. As a signal is decomposed, at first only the high-frequency portions are removed. As the decomposition level increases, the detail components begin to incorporate process dynamics that begin to be filtered along with the noise. To avoid this, a proper threshold must be selected for the signal's reconstruction. Figure 3-13 depicts wavelet-based signal decomposition. ***S*** is the original signal decomposed into its high- and low-frequency components. The figure also shows that by the third decomposition, the detail components seem to contain portions of the signal.

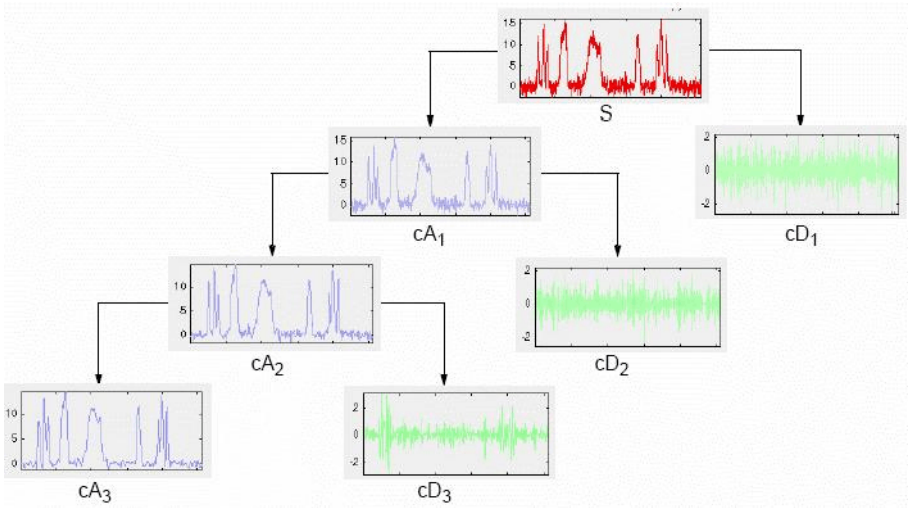


Fig. 3-13. Wavelet signal decomposition.

The detailed components, or noise, are removed from the signal by applying different denoising settings, such as wavelet type, thresholding strategy, and rescaling, to the decomposition [Misiti 2004]. Each parameter combination is then "scored" by testing that the noise estimates are:

1. normally distributed and white,
2. uncorrelated with each other (correlation coefficient under a threshold, ~0.2), and
3. have maximally reduced the variance of the original signal [Miron 2003].

The denoising settings are chosen to optimize a combination of these criteria i to remove process noise and retain process dynamics. If the estimated noise that is removed from each sensor is correlated with one another, one would expect that the correlated portion is actually process dynamics and should not be filtered. The basic assumption is that the true noise is uncorrelated with itself in time and uncorrelated with the noise signals on the other sensors.

3.3.3 Independent component analysis

ICA is a statistical technique that can be used to denoise redundant signal sets. It is most often applied to sensors monitoring safety-critical processes, which usually have high redundancy. ICA expresses the observed data as a linear transformation of latent variables that are mutually independent [Hyvarinen 2001]. ICA has the capability to separate a true process signal (the signal and the common process noise) from the independent channel noise in a group of redundant sensors [Ding 2004]. It is assumed that because most nuclear power plant, safety critical sensors are generally redundant, the ICA technique will perform as well as, or better than, the wavelet denoising technique in estimating true process values.

ICA can be considered a non-Gaussian factor analysis technique that is used to separate independent sources. The central limit theorem (CLT) states that if independent sources are summed, the resulting mixture is more Gaussian than the original sources. To separate the individual sources from the mixture, ICA decomposes the mixture into components that are as non-Gaussian as possible.

The ICA model assumes that p independent sources ($\boldsymbol{S}$) are mixed by a linear mixing matrix ($\boldsymbol{A}$) that results in p measured variables X:

$$\boldsymbol{X} = \boldsymbol{A}\,\boldsymbol{S}\ , \tag{3.3.3.1}$$

where

$\boldsymbol{X}$ is an $\boldsymbol{n} \times \boldsymbol{p}$ data matrix of $\boldsymbol{n}$ observations from $\boldsymbol{p}$ sensors,

$\boldsymbol{S}$ is an $\boldsymbol{n} \times \boldsymbol{p}$ matrix of $\boldsymbol{p}$ independent components,

$\boldsymbol{A}$ is an $\boldsymbol{n} \times \boldsymbol{n}$ matrix of unknown constants, called the mixing matrix [Ding 2004].

The sources ($\boldsymbol{S}$) may be the deterministic portion of the process measurement, common process noise, and independent sensor noise.

The ICA algorithm is used to determine how the variables are mixed. The algorithm returns a constant (weight) matrix, $\boldsymbol{W}$, so that the linear transformation of the observed variables:

$$\boldsymbol{Y} = \boldsymbol{W}\,\boldsymbol{X}\ , \tag{3.3.3.2}$$

results in the transformed components, $\boldsymbol{y_i}$, being as statistically independent from each other as possible. The maximally non-Gaussian signals, $\boldsymbol{Y}$, are then assumed to be estimates of the original independent components. In the case of OLM, one of the independent components is the signal (with common process noise) and thus the optimal parameter estimate.

An ICA algorithm, called FastICA, is commonly used to accomplish this transformation. This algorithm is readily available for use in MATLAB and many other programming languages. The FastICA algorithm uses negentropy $\boldsymbol{J}(\boldsymbol{y})$ as the measurement of the non-Gaussianity of the components. Negentropy is a normalized version of differential entropy. Because the Gaussian distribution is the most random and least structured of all distributions, it will have the largest entropy. Thus, the negentropy value uses the entropy of a Gaussian distribution to gauge the non-Gaussianity of the distribution in question. The negentropy of a distribution is zero only if the distribution is truly Gaussian. Hyvarinen developed FastICA, and his text [2001] provides a complete reference to the algorithm and its theory.

Two ambiguities of ICA are that the variances (energies) and the order of the independent components cannot be determined. These ambiguities are of concern when OLM is being performed because the component containing the parameter estimate needs to be identified and scaled back to its original units. To elaborate, when the ICA algorithm is being executed, a $\boldsymbol{p} \times \boldsymbol{p}$ matrix $\boldsymbol{W}$ is returned. Each of the $\boldsymbol{p}$ rows in $\boldsymbol{W}$ contain the weight values used to construct independent components, with only a single row of weights corresponding to the true signal and common noise sources. The other rows of weights correspond to independent noise. The row of weights in $\boldsymbol{W}$ that contains the true signal and common noise must be distinguished from the other rows. This is not always an easy task. However, Ding [2004] has developed a method of selecting this row based upon the components' correlation to the signals ($\mathbf{X}$). Once the component that contains the parameter estimate is identified, a rescaling method that minimizes the error between the component and the signals ($\boldsymbol{X}$) is implemented.

The major assumptions of the ICA algorithm are that the data are time-invariant and that, at most, only one of the independent components can have a Gaussian distribution. Research has shown that these assumptions generally hold true for nuclear power plant data. Because plants usually operate at nearly

100% power, the assumption of time-invariant data is usually met. Obviously, when the plant undergoes a transient, the data become nonstationary and the ICA method fails. However, transient data are often excluded from OLM estimations. The assumption of a single Gaussian component also is met by most nuclear power operations. This assumption is present because ICA cannot separate Gaussian noise sources. Once multiple Gaussian sources are added they cannot be recovered. Fortunately, with nuclear data, the measurements from each channel contain the process parameter, common noise sources, and independent channel noise sources. Only if the process parameter and an independent noise source were both Gaussian, would ICA fail to yield an estimate of the noise-free signal. Research has validated that the process components and noise components seldom have a Gaussian distribution. Recall that noise sources are commonly assumed to have a Gaussian distribution because when noise sources are added, they tend toward Gaussian; however, this does not imply that the original sources were Gaussian.

3.3.4 Comparison of ICA and wavelet denoising

In this section, the results of a study comparing the filtering capabilities of ICA and wavelet denoising are presented (additional methods are presented in Volume 3 of this series to study the chosen technique's effect on model uncertainty estimation). First, both techniques were used to filter a simulated data set. By using a simulated data set, the true noise statistics were known and used as a benchmark in evaluating the performance of each technique.

For the simulated data set, we considered a source signal $\boldsymbol{s}$ that is described by the following equation:

$$\boldsymbol{s} = \sin\left(2\pi\sqrt{\boldsymbol{t}}\right) . \tag{3.3.4.1}$$

Here, t is uniformly distributed on the interval [0,10].

Three redundant sensors were simulated by introducing a small bias to the actual signal (s) and then adding both a common and an independent noise source. In actual plant operation, common noise could result from the high-frequency boiling of a steam generator, while independent noise could be caused by the electrical contamination of an individual sensor. Here common process noise, $\boldsymbol{N_c}$ (0,0.4), is modeled by drawing random numbers from the Gaussian distribution with a mean of 0 and variance of 0.4. Independent noise, $\boldsymbol{L_i}$ (0,$\boldsymbol{v_i}$) is drawn from the Laplacian distribution with the variable-specific variance $\boldsymbol{v_i}$. These noise sources are added to each channel with differing weights in order to model a contaminated signal measured by a process sensor. Also, by varying the variance of the noise, we can determine if either of the techniques' performance changes significantly with different noise levels. Thus, the three simulated redundant channels are constructed with the following equations:

$$\boldsymbol{X} = \begin{cases} \sin\left(2\pi\sqrt{\boldsymbol{t}}\right) + \boldsymbol{N}_c\left(0,0.4\right) + \boldsymbol{L}_i\left(0,2.4\right) + 1.0 \\ \sin\left(2\pi\sqrt{\boldsymbol{t}}\right) + \boldsymbol{N}_c\left(0,0.4\right) + \boldsymbol{L}_i\left(0,1.6\right) = 0.8 \\ \sin\left(2\pi\sqrt{\boldsymbol{t}}\right) + \boldsymbol{N}_c\left(0,0.4\right) + \boldsymbol{L}_i\left(0,0.8\right) + 1.8 \end{cases} . \tag{3.3.4.2}$$

The parameters used to evaluate performance were

1. the average mean and variance of the estimated noise;
2. the average squared error of the estimated true signal;
3. the average fractional variance reduction, $1 = \frac{\mu(\mathrm{var}(\boldsymbol{X}_s))}{\mu(\mathrm{var}(\boldsymbol{X}))}$ where $\boldsymbol{X}_s$ is the denoised signal, and $\boldsymbol{X}$ is the noisy signal;
4. the average correlation of the noise estimates with each other, $\mu(\boldsymbol{p}_{xy}(\varepsilon_{est}, \varepsilon_{est}))$; and
5. the average correlation of the noise estimate with its smoothed version, $\mu(\boldsymbol{p}_{xy}(\boldsymbol{x}_{i,s}, \varepsilon_{i,est}))$.

The first and second performance metrics (average mean and variance of the estimated noise and expected squared error) measure the accuracy of each denoising method's ability to estimate the noise and the true parameter values, respectively. The next parameter (fractional variance reduction) measures the extent of the denoising. The final two parameters (average noise correlations with themselves and the smoothed variables) measure the denoising method's ability to remove noise that is not correlated with other noise estimates or with the process itself. All of the parameters were averaged over the three sensors to obtain a single estimate. Because ICA is an iterative nonlinear optimization algorithm, which randomly initializes a weight matrix, there is a slight variability in its outcome; even when the same data set is used, different results may be produced each time the algorithm is run. Thus, the results reported for ICA are shown with standard deviations that resulted from multiple trials.

Because the numerical results reported in Table 3-2 are averaged over all of the channels, the results are also depicted graphically in Figs. 3-14 and 3-15, such that each technique's performance may be seen at the unique noise levels. Examination of the figures shows that ICA and wavelet denoising perform consistently over the varied levels. Figure 3-14 also indicates that ICA is effective in removing the large independent noise contributions but is not capable of removing the smaller, base variations. As shown by Fig. 3-15, the situation is reversed for wavelet denoising, which removes most of the base variance but is unable to remove the large spikes. This feature is also illustrated in Table 3-2, where the overall fractional variance reduction is smaller for ICA because it removes the largest noise contributions. Another feature of interest is the correlations of the noise with itself and with its smoothed version. It can be seen that the expected noise correlations for ICA are on the order of ~0.2, which could be argued as being significant and due to filtering process information. This is not the case for the wavelet denoising routine, which produces noise correlations more than an order of magnitude smaller than ICA. Finally, Table 3-2 shows that although there is a slight uncertainty associated with all of the ICA statistics, the uncertainty is not overwhelming. This indicates that the ICA results are fairly repeatable. However, before ICA's smoothed estimate can be used in the uncertainty analysis of any safety critical or high-value operation, it must first be confirmed that the uncertainty contributed by ICA's variability is indeed negligible with respect to the rest of the analysis.

Table 3-2. Denoising comparison for the simulated data set.

	ICA	Wavelet	Actual
Average noise mean	–0.009 ± 0.004	–0.003	–0.004
Average noise variance	2.809 ± 0.457	2.765	2.981
Average squared error	0.278 ± 0.007	0.368	0.000
$1-\frac{\mu(\mathrm{var}(\boldsymbol{X}_1))}{\mu(\mathrm{var}(\boldsymbol{X}))}$	0.837 ± 0.021	0.751	0.762
$\mu(\boldsymbol{p}_{xy}(\varepsilon_{est},\varepsilon_{est}))$	0.079 ± 0.070	0.006	0.010
$\mu(\boldsymbol{p}_{xy}(\varepsilon_{i,?},\varepsilon_{i,est}))$	0.192 ± 0.045	0.013	0.049

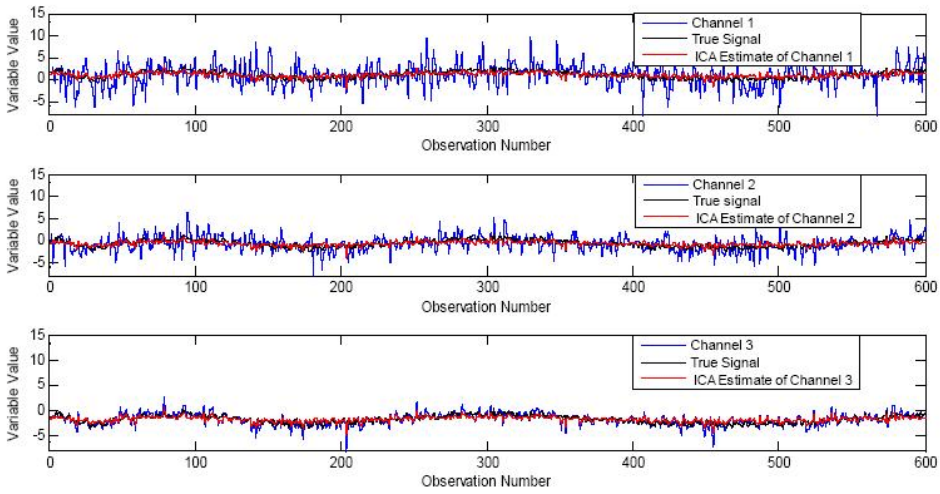


Fig. 3-14. Smoothing estimates for ICA on the simulated data.
(The true signal denotes the signal plus common process noise.)

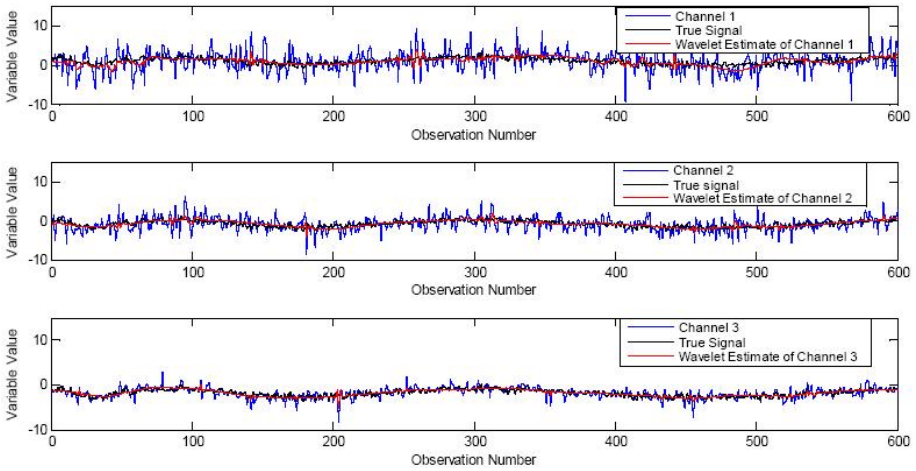


Fig. 3-15. Smoothing estimates for wavelet denoising on simulated data.

Next, both techniques were used to filter actual nuclear power plant data from three redundant steam generator level sensors. Although there is no definite gauge of the technique's performance when using genuine data because the true signal is unknown, it is still useful to apply the techniques to a real-world problem. Figures 3-16 and 3-17 show the results of this application.

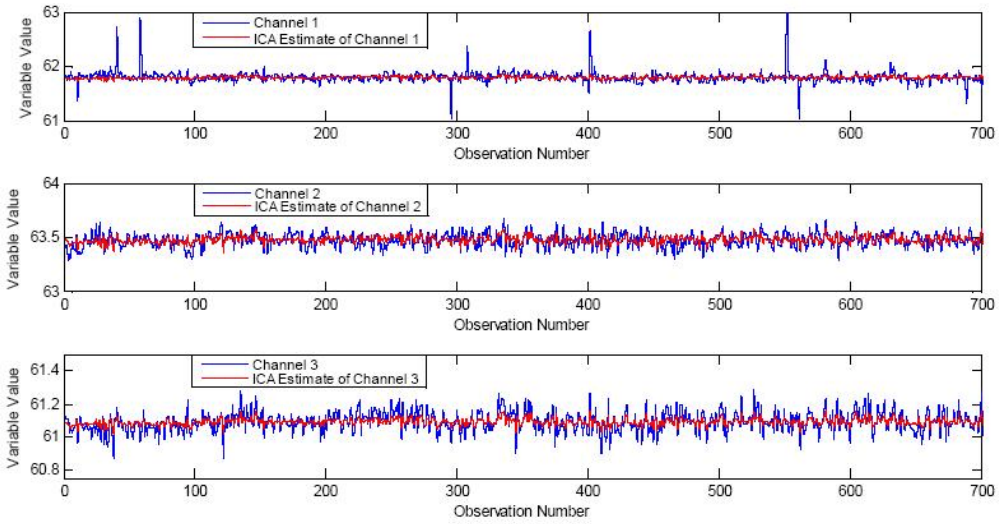


Fig. 3-16. ICA smoothing estimates for the nuclear data set.

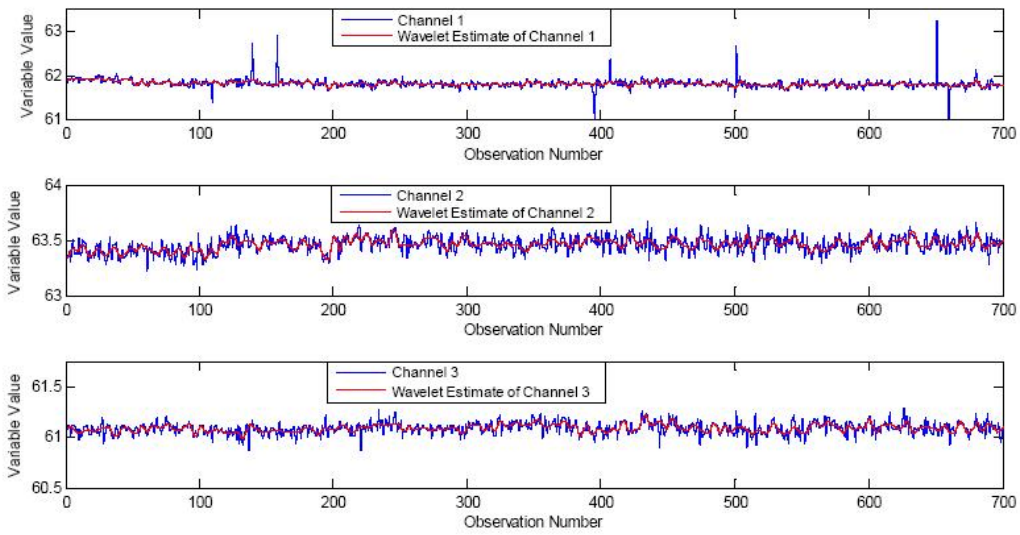


Fig. 3-17. Wavelet denoising smoothing estimates for nuclear data.

Examination of Figs. 3-16 and 3-17 shows that the ICA and wavelet denoising filtering abilities are consistent when applied to real data sets. Again, ICA's smoothed estimate contains more small variations but fewer spikes; whereas the wavelet denoised estimate retains some of the larger independent noise spikes but is successful in removing the base variations. It is also important to note the computational intensity of the wavelet denoising module in the face of large data sets. It took nearly 100 times longer for the wavelet denoising algorithm to execute than the ICA algorithm. This study indicates that wavelet denoising's computational burden could discourage its use. This study also establishes that ICA's requirements of time-invariant data and only a single Gaussian component are met when the technique is applied to nuclear data.

The numerical results of the study using actual data are reported in Table 3-3. Although the true signal and noise statistics are not known, ICA's noise correlations are again much higher than the wavelet correlations, which signify that ICA may be removing a portion of the true process dynamics. However, in the empirical modeling community, correlations with an absolute value less than 0.3 are considered to be slight.

Table 3-3. Denoising comparison for the nuclear stream generator pressure channels.

	ICA	Wavelet
Average noise mean	0.002 ± 0.049	0.001
Average noise variance	0.002 ± 0.000	0.007
$1-\frac{\mu(\mathrm{var}(\boldsymbol{X}_1))}{\mu(\mathrm{var}(\boldsymbol{X}))}$	0.923 ± 0.000	0.553
$\mu(\boldsymbol{p}_{xy}(\varepsilon_{est},\varepsilon_{est}))$	0.303 ± 0.106	0.129
$\mu(\boldsymbol{p}_{xy}(\boldsymbol{X}_{i,?},\varepsilon_{i,est}))$	0.026 ± 0.013	0.056

Overall, the results indicated that both techniques are well-suited for the denoising required in the OLM analytical uncertainty analyses. Both methods seem to yield a reasonable estimate of σ_ε^2. Again, the ICA or wavelet estimate of σ_ε^2 is more robust than those obtained through standard smoothing methods, such as local averaging and median filtering, which exclude most common process noise from the estimate. For the same reason, these methods will also yield better estimates of the bias. Although wavelet denoising appears to be marginally more accurate, ICA may be favored when filtering large data sets because of the large computational requirements of wavelet denoising.

A perceived drawback of ICA is its redundancy requirement. Most nuclear plant sensors are in redundant groups, so obtaining the redundant data is not a problem. However, if used on the secondary side or balance of plant sensors, ICA may not be applicable.

3.4 Uncertainty Summary

In this chapter, the general framework of model predictive uncertainty was presented. The bias-variance decomposition was shown. Also, several methods for bias estimation were explained. These methods computed the bias using a 95th percentile resulting in a conservative estimate, which was in accordance with Requirements 6 and 7 of the SER. Next, several examples comparing the bias calculation methods using simulated data are presented. These examples use data that exhibit sharp curvature to help illustrate the theory behind bias calculation. However, nuclear data being considered for OLM are generally steady state, which means the estimated bias value will be smaller and more constant throughout the process.

This chapter also discussed denoising. ICA and wavelet denoising were highlighted as two viable denoising techniques. Both of these techniques have the theoretical ability to differentiate between independent noise and common noise. ICA and wavelet denoising were compared using both artificially simulated data and real nuclear data. This comparison showed that both methods produce acceptable denoising results, which are needed for OLM uncertainty analyses. Chapters 5 and 6 will present analytical and Monte Carlo-based techniques, respectively, for performing these OLM uncertainty analyses on the AAKR, AAMSET, and AANN models.

4. ANALYTIC METHODS

This section will present the derivation of the analytic equations for the three empirical models currently available in OLM systems, specifically AANN, AAKR, and AAMSET.

4.1 Auto-Associative Neural Networks

Recall that an auto-associative neural network (AANN) is a parametric modeling technique composed of a four-layer neural network. The layers of an AANN act to map data onto nonlinear components (the mapping layer), perform a nonlinear transformation on the components (the bottleneck layer), de-map these transformations (the de-mapping layer), and then combine these components into predictions (the output layer). This network architecture is illustrated in Fig. 4-1. In addition, the symbols used to represent the different artificial neurons may be seen in Fig. 4-2. This section will not provide an in-depth general discussion of neural network architectures, training algorithms, or optimization techniques. A number of well written texts are dedicated to this subject, one of which is *Neural Networks: A Comprehensive Foundation* by Haykin [1994].

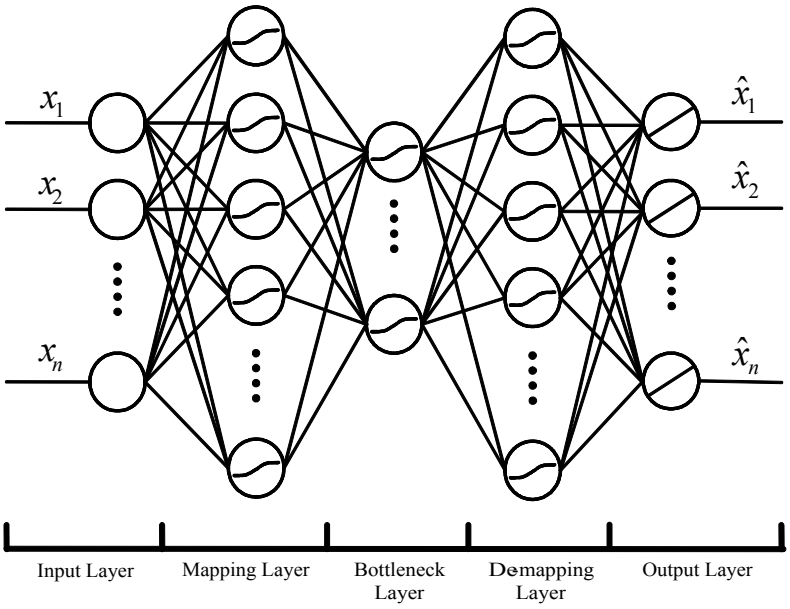


Fig. 4-1. Auto-associative neural network architecture.

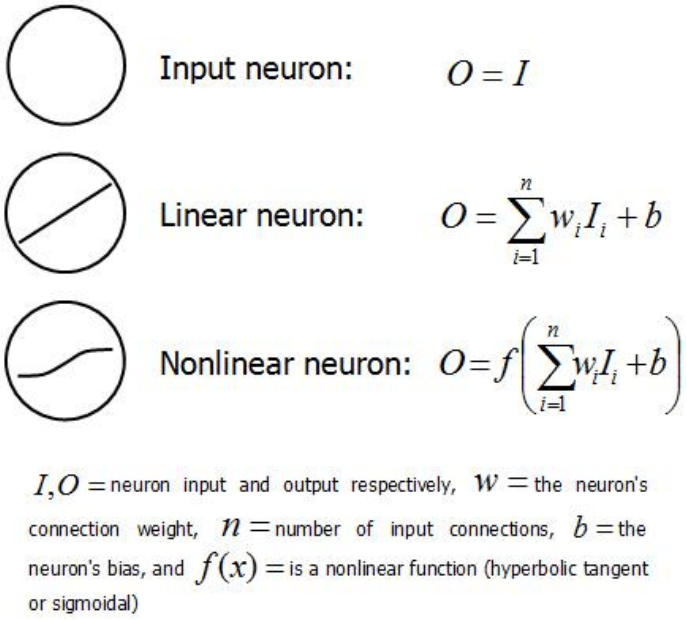


Fig. 4-2. Artificial neuron symbol definitions.

4.1.1 Uncertainty intervals for a general nonlinear regression model

This section will present a modified form of the derivation presented by Rasmussen [2003] for confidence and prediction interval estimation for general nonlinear regression models. Consider the following nonlinear system being modeled:

$$\boldsymbol{Y}_i = \boldsymbol{f}\left(\mathrm{X}_i, \theta\right) + \varepsilon_i \ , \tag{4.1.1.1}$$

where

$\boldsymbol{Y}_i$ is the measured response variable;

$\mathrm{X}_i = \left[\boldsymbol{X}_{i,1} \quad \boldsymbol{X}_{i,2} \quad \cdots \quad \boldsymbol{X}_{i,p}\right]$ are the $\boldsymbol{p}$ predictor variables of $\boldsymbol{Y}_i$,

è are the model parameters,

$\boldsymbol{f}(\boldsymbol{X}_i, \boldsymbol{è})$ is the true value of the response variable for a given $\boldsymbol{X}_i$ and $\boldsymbol{\theta}$,

$\varepsilon_i \approx \boldsymbol{N}\left(0, \sigma_\varepsilon^2\right)$ is normally distributed random noise with a mean 0 and variance σ_ε^2,

$\boldsymbol{I} = 1,2,\ldots,\boldsymbol{n}_{trn}$ are the indices of the training observations with $\boldsymbol{n}_{trn}$ being the number of training observations.

The sum of the squared errors is thus given by:

$$\boldsymbol{S}(\grave{\mathbf{e}}) = \sum_{i=1}^{n_m} \left[\boldsymbol{Y}_i - \boldsymbol{f}(\mathrm{X}_i, \grave{\mathbf{e}}) \right]^2 \quad . \tag{4.1.1.2}$$

At this point, $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ is defined as the set of model parameters that minimizes $\boldsymbol{S}(\boldsymbol{\theta})$, $\boldsymbol{P}$ is defined as the number of model parameters or the length of $\hat{\boldsymbol{\theta}}$, and *x* designates a single query observation.

The above function may now be approximated by a first-order Taylor polynomial of $\boldsymbol{\theta}$ about $\hat{\boldsymbol{\theta}}$:

$$\boldsymbol{f}(\mathbf{x}, \hat{\boldsymbol{\theta}}) \approx \boldsymbol{f}(\mathbf{x}, \boldsymbol{\theta}_0) + \sum_{k=1}^{P} \left[\left(\frac{\partial \boldsymbol{f}(\mathbf{x}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}_k} \right) \left(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k \right) \right] \quad . \tag{4.1.1.3}$$

Using the definition of a Jacobian gradient vector, which is the transpose of the gradient vector **f**,

$$\mathrm{f}^T = \left[\frac{\partial \boldsymbol{f}(\mathrm{x}, \hat{\mathbf{e}})}{\partial \grave{\mathbf{e}}_1} \frac{\partial \boldsymbol{f}(\mathrm{x}, \hat{\mathbf{e}})}{\partial \grave{\mathbf{e}}_2} \cdots \frac{\partial \boldsymbol{f}(\mathrm{x}, \hat{\mathbf{e}})}{\partial \grave{\mathbf{e}}_p} \right] \quad . \tag{4.1.1.4}$$

The following equation results:

$$\boldsymbol{f}(\mathrm{x}, \hat{\mathbf{e}}) \approx \boldsymbol{f}(\mathrm{x}, \grave{\mathbf{e}}) + \mathrm{f}^T (\hat{\mathbf{e}} - \grave{\mathbf{e}}) \quad . \tag{4.1.1.5}$$

The variance of $\boldsymbol{f}(\mathrm{x}, \hat{\theta})$, is given by Dybowski [1997] to be

$$\boldsymbol{Var}\left[\boldsymbol{f}(\mathrm{x}, \hat{\theta}) \right] = \mathrm{f}^T \mathrm{S}\, \mathrm{f} \quad . \tag{4.1.1.6}$$

The variance covariance matrix, ***S***, may be estimated in several ways using an estimated Hessian matrix [Dybowski 1997, Tibshirani 1996], but since estimating the Hessian is usually difficult, if indeed possible, the following estimate will be used [Chryssolouris 1996]:

$$\mathrm{S} = \boldsymbol{s}^2 \left[\mathrm{F}^T \mathrm{F} \right]^{-1} \quad . \tag{4.1.1.7}$$

where

F is an $\boldsymbol{n}_{trn} \times \boldsymbol{P}$ matrix of partial derivatives of $\boldsymbol{f}(\mathbf{X}_i, \hat{\boldsymbol{\theta}})$ with respect to the model parameters determined from the least squares minimization:

that is, $\mathbf{F}_{i,j} = \dfrac{\partial \boldsymbol{f}(\mathbf{X}_i, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}_j}$;

$\boldsymbol{s}^2 = \frac{1}{\boldsymbol{n}_{trn}} \sum_{i=1}^{n_{trn}} \left[\boldsymbol{Y}_i - \boldsymbol{f}\left(\mathrm{x}_i, \hat{\theta}\right) \right]^2$ is an estimate of the noise variance [Tibshirani 1996].

Now consider the nonlinear model for a new query observation $\boldsymbol{x}$:

$$\boldsymbol{y} = \boldsymbol{f}\left(\mathrm{x}, \theta\right) + \varepsilon \ . \tag{4.1.1.8}$$

The model's estimate of the response is given by:

$$\hat{\boldsymbol{y}} = \boldsymbol{f}\left(\mathrm{x}, \hat{\theta}\right) \ . \tag{4.1.1.9}$$

Calculating the difference between the estimate and the measured response:

$$\boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{f}\left(\mathrm{x}, \theta\right) + \varepsilon - \boldsymbol{f}\left(\mathrm{x}, \hat{\theta}\right) \ . \tag{4.1.1.10}$$

Recall that the first-order expansion of the model's estimate of $\boldsymbol{\theta}$ about $\hat{\boldsymbol{\theta}}$ is given by:

$$\boldsymbol{f}\left(\mathrm{x}, \hat{\theta}\right) \approx \boldsymbol{f}\left(\mathrm{x}, \theta\right) + \mathrm{f}^T\left(\hat{\theta} - \theta\right) \ . \tag{4.1.1.11}$$

Then the above difference equation may be approximated by:

$$\begin{aligned} \boldsymbol{y} - \hat{\boldsymbol{y}} &\approx \boldsymbol{f}\left(\mathrm{x}, \theta\right) + \varepsilon - \boldsymbol{f}\left(\mathrm{x}, \theta\right) - \mathrm{f}^T\left(\hat{\theta} - \theta\right) \ , \\ \boldsymbol{y} - \hat{\boldsymbol{y}} &\approx \varepsilon = \mathrm{f}^T\left(\hat{\theta} - \theta\right) \ . \end{aligned} \tag{4.1.1.12}$$

It is assumed that ε and $\mathbf{f}^T\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right)$ are independent. This results in the following variance decomposition:

$$\begin{aligned} \boldsymbol{Var}\left[\boldsymbol{y} - \hat{\boldsymbol{y}}\right] &= \boldsymbol{Var}\left[\varepsilon - \mathrm{f}^T\left(\hat{\theta} - \theta\right)\right], \\ \boldsymbol{Var}\left[\boldsymbol{y} - \hat{\boldsymbol{y}}\right] &= \boldsymbol{Var}\left[\varepsilon\right] + \boldsymbol{Var}\left[\mathrm{f}^T\left(\hat{\theta} - \theta\right)\right] \ . \end{aligned} \tag{4.1.1.13}$$

Recall that $\varepsilon \approx \boldsymbol{N}(0, \sigma_\varepsilon^2)$ (i.e., normally distributed with a mean of 0 and variance of σ_ε^2) and that the variance of the prediction is dependent on the difference between the optimum set of model parameters $\hat{\boldsymbol{\theta}}$ and the current set of model parameters $\boldsymbol{\theta}$ that is assumed to be normally distributed, mainly $\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right) \approx \boldsymbol{N}(0, \mathbf{S})$. The variance of the difference now becomes

$$\boldsymbol{Var}\left[\boldsymbol{y}-\hat{\boldsymbol{y}}\right]\approx\sigma_{\varepsilon}^{2}+\mathrm{f}^{T}\mathrm{S}\,\mathrm{f}\ . \tag{4.1.1.14}$$

Using the estimate of $\boldsymbol{S}$ defined by Chryssolouris [1996]:

$$\boldsymbol{Var}\left[\boldsymbol{y}-\hat{\boldsymbol{y}}\right]\approx\sigma_{\varepsilon}^{2}+\mathrm{f}^{\boldsymbol{T}}\boldsymbol{s}^{2}\left[\mathrm{F}^{\boldsymbol{T}}\mathrm{F}\right]^{-1}\mathrm{f}\ , \tag{4.1.1.15}$$

recall that the variance, σ_{ε}^{2}, is estimated by the mean squared training error $\boldsymbol{s}^{2}$, *so*

$$\boldsymbol{Var}\left[\boldsymbol{y}-\hat{\boldsymbol{y}}\right]\approx\boldsymbol{s}^{2}+\boldsymbol{s}^{2}\left(\mathrm{f}^{\boldsymbol{T}}\left[\mathrm{F}^{\boldsymbol{T}}\mathrm{F}\right]^{-1}\mathrm{f}\right)\ . \tag{4.1.1.16}$$

The Student's **t**-distribution is then estimated by:

$$\boldsymbol{t}_{\boldsymbol{n}_{trn}-\boldsymbol{p}}\approx\frac{\boldsymbol{y}-\hat{\boldsymbol{y}}}{\sqrt{\boldsymbol{Var}\left[\boldsymbol{y}-\hat{\boldsymbol{y}}\right]}}\approx\frac{\boldsymbol{y}-\hat{\boldsymbol{y}}}{\boldsymbol{s}\sqrt{1+\mathbf{f}^{\boldsymbol{T}}\left[\mathbf{F}^{\boldsymbol{T}}\mathbf{F}\right]^{-1}\mathbf{f}}}\ . \tag{4.1.1.17}$$

Here, $\boldsymbol{n}_{\boldsymbol{trn}}-\boldsymbol{p}$ is the number of degrees of freedom for the distribution, with $\boldsymbol{n}_{\boldsymbol{trn}}$ being the number of training observations and $\boldsymbol{p}$ being the number of variables used to estimate y.

Therefore, the associated prediction and its corresponding prediction interval for a theoretical coverage of $1-\alpha$ is found to be:

$$\boldsymbol{E}(\hat{\boldsymbol{y}})\pm\boldsymbol{t}_{\boldsymbol{n}_{trn}-\boldsymbol{p},\alpha/2}\ \boldsymbol{s}\sqrt{1+\mathbf{f}^{\boldsymbol{T}}\left[\mathbf{F}^{\boldsymbol{T}}\mathbf{F}\right]^{-1}\mathbf{f}}\ . \tag{4.1.1.18}$$

Before continuing, recall that Eq. (4.1.1.18) is for the prediction interval of an unbiased model. To construct a proper prediction interval, as described in Sect. 3.2.2 a bias term must be added as follows:

$$\hat{\boldsymbol{y}}\pm\boldsymbol{t}_{\boldsymbol{n}_{trn}-\boldsymbol{p},\alpha/2}\ \sqrt{\boldsymbol{s}^{2}+\boldsymbol{s}^{2}\ \mathbf{f}^{\boldsymbol{T}}\left[\mathbf{F}^{\boldsymbol{T}}\mathbf{F}\right]^{-1}\mathbf{f}+\boldsymbol{Bias}^{2}}\ . \tag{4.1.1.19}$$

Because alternative methods exist for estimating the variance of the noise (see Sect. 3.4.), a more generic term for the estimated noise variance, $\hat{\sigma}_{\varepsilon}^{2}$, may be introduced:

$$\hat{\boldsymbol{y}}\pm\boldsymbol{t}_{\boldsymbol{n}_{trn}-\boldsymbol{p},\alpha/2}\ \sqrt{\hat{\sigma}_{\varepsilon}^{2}+\hat{\sigma}_{\varepsilon}^{2}\ \mathbf{f}^{\boldsymbol{T}}\left[\mathbf{F}^{\boldsymbol{T}}\mathbf{F}\right]^{-1}\mathbf{f}+\boldsymbol{Bias}^{2}}\ . \tag{4.1.1.20}$$

Reexamination of Eqs. (3.2.2.4) and (3.2.2.5) shows that the prediction interval and confidence interval are the same except for the fact that the confidence interval does not contain a noise term $\left(\sigma_{\varepsilon}^{2}\right)$. Thus, a confidence interval can be derived from Eq. (4.1.1.19) by neglecting the estimated noise variance, $\hat{\sigma}_{\varepsilon}^{2}$. Thus, a confidence interval is constructed with Eq. (4.1.1.20):

$$E(\hat{y}) \pm t_{n_{trn}-p,\alpha/2} \sqrt{\hat{\sigma}_\varepsilon^2 \, \mathbf{f}^T \left[\mathbf{F}^T\mathbf{F}\right]^{-1} \mathbf{f} + Bias^2} \quad . \tag{4.1.1.21}$$

The terms of Eqs. (4.1.1.20) and (4.1.1.21) will now be derived for an AANN.

4.1.2 Uncertainty intervals for auto-associative neural networks

This section will use the approach developed for a general nonlinear regression model to estimate the uncertainty for the predictions made by an AANN. The discussion presented below follows the general approach presented by Rasmussen [2003] for an inferential artificial neural network (IANN), where the IANN uses *p* variables to estimate a single response variable. The difference in the following discussion is the extension of this approach to each of the AANN's predicted variables. In other words, the AANN is broken into *p* IANNs where the *p* predictor variables are used to estimate the *i*th response variable for $i = \{1, 2, \ldots, p\}$.

First, recall the following expression for the $1-\alpha$ prediction interval of a general, un-biased nonlinear regression model:

$$\hat{y} \pm t_{n_{trn}-p,\alpha/2} \, s\sqrt{1 + \mathbf{f}^T \left[\mathbf{F}^T\mathbf{F}\right]^{-1} \mathbf{f}} \quad . \tag{4.1.2.1}$$

The uncertainty of the estimate with a certainty of $1-\alpha$ is given by:

$$t_{n_{trn}-p,\alpha/2} \, s\sqrt{1 + \mathbf{f}^T \left[\mathbf{F}^T\mathbf{F}\right]^{-1} \mathbf{f}} \quad . \tag{4.1.2.2}$$

Recall that **F** is defined as the $n_{trn} \times P$ Jacobian matrix of the model's nonlinear function with respect to each of the modeling parameters estimated in the training process. Here, n_{trn} is the number of training observations, and P is the number of parameters in the model. Also, recall that **f** is the $1 \times P$ Jacobian matrix of the model's nonlinear function with respect to its parameters for a query observation *x*; that is,

$$\mathbf{f}^T = \left[\frac{\partial f\left(\mathbf{x}, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_1} \quad \frac{\partial f\left(\mathbf{x}, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_2} \quad \ldots \quad \frac{\partial f\left(\mathbf{x}, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_P} \right] . \tag{4.1.2.3}$$

The nominal response uncertainty estimate is given by

$$s^2 = \frac{1}{n_{trn}} \sum_{i=1}^{n_{trn}} \left[Y_i - f\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right) \right]^2 \quad . \tag{4.1.2.4}$$

At first glance, it is not difficult to recognize the terms that require in-depth consideration, namely **F** and **f**. The majority of this section will address the derivation of these two quantities and will extend their equations to an AANN.

The architecture of a general AANN (Fig. 4-3) can be treated as a set of isolated, multilayer IANNs. One such isolation is shown by the green lines in Fig. 4-3, neglecting the black lines which make the network auto-associative. In this diagram each of the neuron bias values has been omitted for the sake of simplicity.

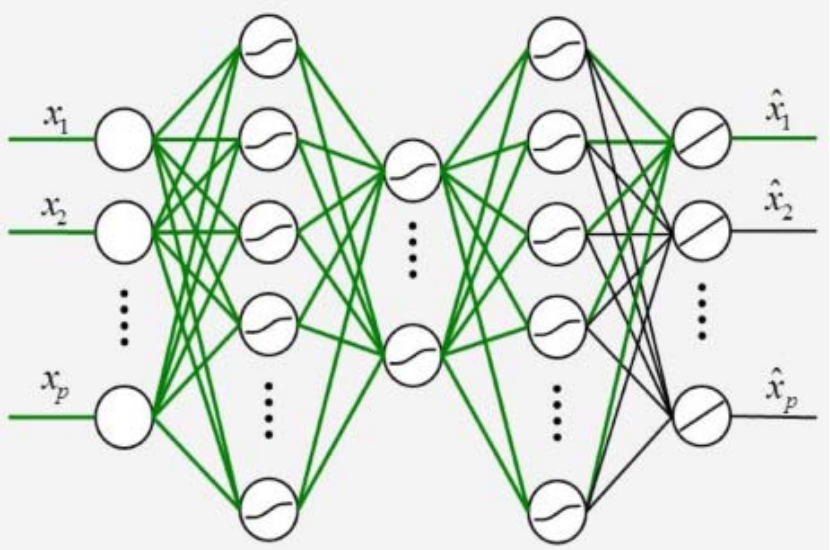


Fig. 4-3. Example decomposition of AANN into multilayer IANN.

The derivation will now be presented with respect to one such isolated IANN contained within the AANN architecture.

The first step is the determination of the dimensions of **F** for an isolated IANN. Define $\boldsymbol{p}$ as the number of input/output neurons (i.e., number of variables being monitored by the AANN), $\boldsymbol{n_m}$ as the number of mapping/demapping neurons, and $\boldsymbol{n_b}$ as the number of bottleneck neurons. The breakdown of the number of variables for each isolated IANN's layer connections may be seen in Table 4-1.

Table 4-1. Number of parameters in multilayer IANN

	Number of biases	Number of weights
Input layer → mapping layer	n_m	$p \; x \; n_m$
Mapping layer → bottleneck layer	n_b	$n_m x \; n_b$
Bottleneck layer → demapping layer	n_m	$n_b x \; n_m$
Demapping layer → output layer	1	n_m

Combining these contributions:

$$\boldsymbol{P} = \left(\boldsymbol{n_m} + \boldsymbol{p} \times \boldsymbol{n_m}\right) + \left(\boldsymbol{n_b} + \boldsymbol{n_m} \times \boldsymbol{n_b}\right) + \left(\boldsymbol{n_m} + \boldsymbol{n_m} \times \boldsymbol{n_b}\right) + \left(1 + \boldsymbol{n_m}\right)$$

$$\boldsymbol{P} = 3\boldsymbol{n_m} + \boldsymbol{n_b} + \left(\boldsymbol{p} \times \boldsymbol{n_m}\right) + 2\left(\boldsymbol{n_m} \times \boldsymbol{n_b}\right) + 1 \qquad (4.1.2.5)$$

Therefore, the Jacobian of the IANN for $\boldsymbol{n_{trn}}$ training observations will have dimensions $\boldsymbol{n_{trn}} \times \boldsymbol{P}$ and is presented below, in terms of the generic $\boldsymbol{\theta_{i,j}}$ parameters, where $\boldsymbol{i}$ is the training observation number, and $\boldsymbol{j}$ is the model parameter number:

$$\mathbf{F} = \begin{bmatrix} \frac{\partial f\left(\mathbf{X}_1, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{1,1}} & \frac{\partial f\left(\mathbf{X}_1, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{1,2}} & \cdots & \frac{\partial f\left(\mathbf{X}_1, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{1,P}} \\ \frac{\partial f\left(\mathbf{X}_2, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{2,1}} & \frac{\partial f\left(\mathbf{X}_2, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{2,2}} & \cdots & \frac{\partial f\left(\mathbf{X}_2, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{2,P}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f\left(\mathbf{X}_{n_{trn}}, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{n_{trn},1}} & \frac{\partial f\left(\mathbf{X}_{n_{trn}}, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{n_{trn},2}} & \cdots & \frac{\partial f\left(\mathbf{X}_{n_{trn}}, \hat{\boldsymbol{\theta}}\right)}{\partial \boldsymbol{\theta}_{n_{trn},P}} \end{bmatrix}. \tag{4.1.2.6}$$

Now the question arises: how are these generic parameters related to the weight and bias values of an IANN? Note that the calculation of the above Jacobian is not a new concept, but is calculated for use in the Levenberg-Marquardt neural network training algorithms [Demuth and Beale 2004].

First, a generic weight and bias Jacobian is defined for the L layer of a neural network, training observation i, and k response variable or output neuron.

$$\mathbf{W}_i^L = \begin{bmatrix} \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial \mathbf{w}_1^L} & \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial \mathbf{w}_2^L} & \cdots & \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial \mathbf{w}_{n_L}^L} \end{bmatrix}. \tag{4.1.2.7}$$

Where

n_L and n_{L-1} are the number of neurons in layer L and L – 1, respectively;

$\hat{\boldsymbol{\theta}}$ is the complete set of trained network parameters for the IANN;

$\mathbf{w}_j^L = \begin{bmatrix} w_{1,j}^L & w_{2,j}^L & \cdots & w_{n_{L-1},j}^L \end{bmatrix}$ are the connection weights from neuron j in layer L to all of the neurons in the previous layer, i.e., L – 1;

$\frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial \mathbf{w}_k^L} = \begin{bmatrix} \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial w_{1,k}^L} & \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial w_{2,k}^L} & \cdots & \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial w_{n_{L-1},k}^L} \end{bmatrix}$ is the partial derivative of the IANN prediction for the kth response variable with respect to each of the input weights;

$w_{i,j}^L$ the connection weight of neuron i in layer L – 1 to neuron j in layer L.

$$\mathbf{b}_i^L = \begin{bmatrix} \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial b_1^L} & \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial b_2^L} & \cdots & \frac{\partial \hat{x}_k\left(\mathbf{X}_i, \hat{\boldsymbol{\theta}}\right)}{\partial b_{n_L}^L} \end{bmatrix}, \tag{4.1.2.8}$$

where $\mathbf{b}_i^L = \begin{bmatrix} b_1^L & b_2^L & \cdots & b_{n_L}^L \end{bmatrix}$ are the bias values for the n_L neurons is layer L, and

b_j^L is the bias of neuron j in layer L.

With these definitions, **F** may be written as:

$$\mathbf{F} = \begin{bmatrix} \mathbf{W}_1^1 & \mathbf{b}_1^1 & \mathbf{W}_1^2 & \mathbf{b}_1^2 & \cdots & \mathbf{W}_1^4 & \mathbf{b}_1^4 \\ \mathbf{W}_2^1 & \mathbf{b}_2^1 & \mathbf{W}_2^2 & \mathbf{b}_2^2 & \cdots & \mathbf{W}_2^4 & \mathbf{b}_2^4 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \mathbf{W}_{n_{trn}}^1 & \mathbf{b}_{n_{trn}}^1 & \mathbf{W}_{n_{trn}}^2 & \mathbf{b}_{n_{trn}}^2 & \cdots & \mathbf{W}_{n_{trn}}^4 & \mathbf{b}_{n_{trn}}^4 \end{bmatrix} . \tag{4.1.2.9}$$

Now that the general structure of the Jacobian has been presented, the derivation of its components will be discussed. The foundation of this analysis is taken from the following representation of a single neuron of the isolated IANN, described earlier (Fig. 4-4).

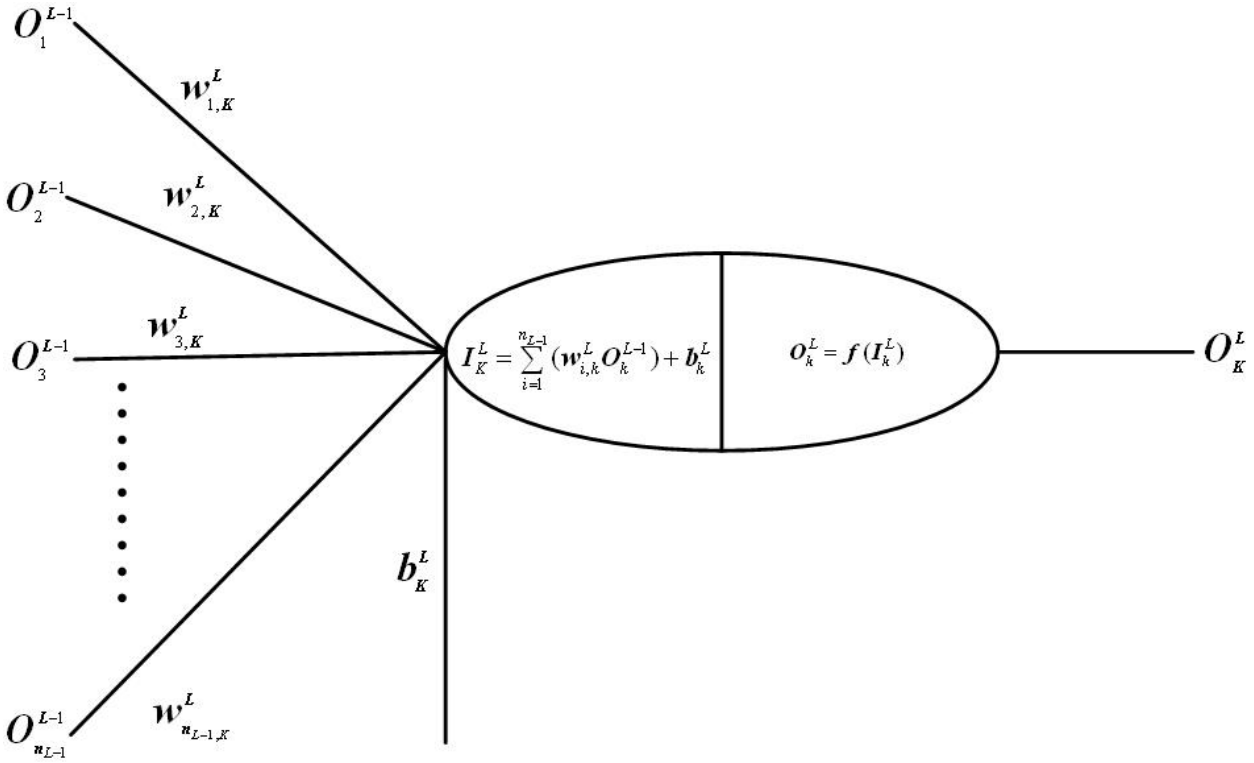


Fig. 4-4. Illustration of neuron *K* in the *L* layer of an isolated IANN.

Here $\boldsymbol{I}_{\boldsymbol{K}}^{\boldsymbol{L}}$ is the internal activation of the neuron, and $\boldsymbol{O}_{\boldsymbol{K}}^{\boldsymbol{L}}$ is the neuron's output. Therefore, the determination of the Jacobian is accomplished by differentiating the network's output with respect to all of the network weight and bias values.

To start, the output of the ***k***th neuron in the output layer will be examined. Recall that the transfer functions of the output neurons are linear and that for this piecewise consideration of the AANN, only one output neuron is present. Because the internal activation is equal to the output for a linear neuron, the neuron's output may be written as follows:

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \boldsymbol{O}_{\boldsymbol{k}}^4 = \sum_{i=1}^{n_m} \left(\boldsymbol{w}_{i,k}^4 \cdot \boldsymbol{O}_i^3 \right) + \boldsymbol{b}_k^4 \quad . \tag{4.1.2.10}$$

Computing the partial derivatives with respect to the neuron's input weights, where $\boldsymbol{i} = 1, 2, \ldots, \boldsymbol{n_m}$ and n_m is the number of neurons in the mapping and demapping layer:

$$\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{w}_{i,k}^4} = \frac{\partial \boldsymbol{O}_k^4}{\partial \boldsymbol{w}_{i,k}^4} = \frac{\partial}{\partial \boldsymbol{w}_{i,k}^4}\left(\boldsymbol{w}_{1,k}^4 \boldsymbol{O}_1^3 + \cdots + \boldsymbol{w}_{i,k}^4 \boldsymbol{O}_i^3 + \cdots + \boldsymbol{w}_{n_m,k}^4 \boldsymbol{O}_{n_m}^3 + \boldsymbol{b}_k^4\right)$$

$$\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{w}_{i,k}^4} = \frac{\partial \boldsymbol{O}_k^4}{\partial \boldsymbol{w}_{i,k}^4} = \boldsymbol{O}_i^3 . \qquad (4.1.2.11)$$

Computing the partial derivatives with respect to the neuron's bias value:

$$\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{b}_k^4} = \frac{\partial \boldsymbol{O}_k^4}{\partial \boldsymbol{b}_k^4} = \frac{\partial}{\partial \boldsymbol{b}_k^4}\left(\boldsymbol{w}_{1,k}^4 \boldsymbol{O}_1^3 + \cdots + \boldsymbol{w}_{i,k}^4 \boldsymbol{O}_i^3 + \cdots + \boldsymbol{w}_{n_m,k}^4 \boldsymbol{O}_{n_m}^3 + \boldsymbol{b}_k^4\right)$$

$$\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{b}_k^4} = \frac{\partial \boldsymbol{O}_k^4}{\partial \boldsymbol{b}_k^4} = 1 \qquad (4.1.2.12)$$

Moving to the third layer, the output of the mth neuron is given by the following equation, where $\boldsymbol{i} = 1, 2, \ldots, \boldsymbol{n_b}$, and $\boldsymbol{n_b}$ is the number of neurons in the bottleneck layer:

$$\boldsymbol{O}_m^3 = \boldsymbol{f}\left(\sum_{i=1}^{n_b}\left(\boldsymbol{w}_{i,m}^3 \cdot \boldsymbol{O}_i^2\right) + \boldsymbol{b}_m^3\right) = \tanh\left(\sum_{i=1}^{n_b}\left(\boldsymbol{w}_{i,m}^3 \cdot \boldsymbol{O}_i^2\right) + \boldsymbol{b}_m^3\right) . \qquad (4.1.2.13)$$

Note that $\tanh(\boldsymbol{x}) = \frac{\boldsymbol{e}^{2x} - 1}{\boldsymbol{e}^{2x} + 1}$ and $\boldsymbol{I}_m^3 = \sum_{i=1}^{n_b}\left(\boldsymbol{w}_{i,m}^3 \cdot \boldsymbol{O}_i^2\right) + \boldsymbol{b}_m^3$.

$$\boldsymbol{O}_m^3 = \frac{2}{1 + \boldsymbol{e}^{-2\left(\boldsymbol{I}_m^3\right)}} - 1 . \qquad (4.1.2.14)$$

For this layer, the differentiations are slightly more complicated and involve the decomposition of $\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{w}_{i,m}^3}$ and $\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{b}_m^3}$ using the chain rule as follows:

$$\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{w}_{i,m}^3} = \left(\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{I}_{m,k}^4}\right) \cdot \left(\frac{\partial \boldsymbol{I}_{m,k}^4}{\partial \boldsymbol{O}_m^3}\right) \cdot \left(\frac{\partial \boldsymbol{O}_m^3}{\partial \boldsymbol{I}_m^3}\right) \cdot \left(\frac{\partial \boldsymbol{I}_m^3}{\partial \boldsymbol{w}_{i,m}^3}\right)$$

$$\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{b}_m^3} = \left(\frac{\partial \hat{\boldsymbol{x}}_k}{\partial \boldsymbol{I}_{m,k}^4}\right) \cdot \left(\frac{\partial \boldsymbol{I}_{m,k}^4}{\partial \boldsymbol{O}_m^3}\right) \cdot \left(\frac{\partial \boldsymbol{O}_m^3}{\partial \boldsymbol{I}_m^3}\right) \cdot \left(\frac{\partial \boldsymbol{I}_m^3}{\partial \boldsymbol{b}_m^3}\right) \qquad (4.1.2.15)$$

Here, $\boldsymbol{I}_{m,k}^{4}$ refers to the *m*th neuron's contribution to the internal activation of the *k*th output neuron, that is, $\boldsymbol{I}_{m,k}^{4} = \boldsymbol{w}_{m,k}^{4} \cdot \boldsymbol{O}_{m}^{3}$.

The various components of the above derivatives must now be determined. Note that each of the two derivative equations contain $\frac{\partial \hat{\boldsymbol{x}}_{k}}{\partial \boldsymbol{I}_{m,k}^{4}}$, $\frac{\partial \boldsymbol{I}_{m,k}^{4}}{\partial \boldsymbol{O}_{m}^{3}}$, and $\frac{\partial \boldsymbol{O}_{m}^{3}}{\partial \boldsymbol{I}_{m}^{3}}$. These are determined to be

$$\frac{\partial \hat{\boldsymbol{x}}_{k}}{\partial \boldsymbol{I}_{m,k}^{4}} = \frac{\partial}{\partial \boldsymbol{I}_{m,k}^{4}}\left(\boldsymbol{O}_{m,k}^{4}\right) = \frac{\partial}{\partial \boldsymbol{I}_{m,k}^{4}}\left(\boldsymbol{I}_{m,k}^{4}\right) = 1$$

$$\frac{\partial \boldsymbol{I}_{m,k}^{4}}{\partial \boldsymbol{O}_{m}^{3}} = \frac{\partial}{\partial \boldsymbol{O}_{m}^{3}}\left(\boldsymbol{w}_{m,k}^{4} \cdot \boldsymbol{O}_{m}^{3}\right) = \boldsymbol{w}_{m,k}^{4} \quad . \tag{4.1.2.16}$$

$$\frac{\partial \boldsymbol{O}_{m}^{3}}{\partial \boldsymbol{I}_{m}^{3}} = \frac{\partial}{\partial \boldsymbol{I}_{m}^{3}}\left(\frac{\boldsymbol{e}^{2\boldsymbol{I}_{m}^{3}} - 1}{\boldsymbol{e}^{2\boldsymbol{I}_{m}^{3}} + 1}\right) = 1 - \left(\boldsymbol{O}_{m}^{3}\right)^{2}$$

This leaves $\frac{\partial \boldsymbol{I}_{m}^{3}}{\partial \boldsymbol{w}_{i,m}^{3}}$ and $\frac{\partial \boldsymbol{I}_{m}^{3}}{\partial \boldsymbol{b}_{m}^{3}}$:

$$\frac{\partial \boldsymbol{I}_{m}^{3}}{\partial \boldsymbol{w}_{i,m}^{3}} = \frac{\partial}{\partial \boldsymbol{w}_{i,m}^{3}}\left(\sum_{i=1}^{n_b}\left(\boldsymbol{w}_{i,m}^{3} \cdot \boldsymbol{O}_{i}^{2}\right) + \boldsymbol{b}_{m}^{3}\right) = \boldsymbol{O}_{i}^{2}$$

$$\frac{\partial \boldsymbol{I}_{m}^{3}}{\partial \boldsymbol{b}_{m}^{3}} = \frac{\partial}{\partial \boldsymbol{b}_{m}^{3}}\left(\sum_{i=1}^{n_b}\left(\boldsymbol{w}_{i,m}^{3} \cdot \boldsymbol{O}_{i}^{2}\right) + \boldsymbol{b}_{m}^{3}\right) = 1 \quad . \tag{4.1.2.17}$$

Combining these partial derivatives to find the final form of the Jacobian entries for the third layer are found to be:

$$\frac{\partial \hat{\boldsymbol{x}}_{k}}{\partial \boldsymbol{w}_{i,m}^{3}} = (1) \cdot \left(\boldsymbol{w}_{m,k}^{4}\right) \cdot \left(1 - \left(\boldsymbol{O}_{m}^{3}\right)^{2}\right) \cdot \left(\boldsymbol{O}_{i}^{2}\right)$$

$$\frac{\partial \hat{\boldsymbol{x}}_{k}}{\partial \boldsymbol{b}_{m}^{3}} = (1) \cdot \left(\boldsymbol{w}_{m,k}^{4}\right) \cdot \left(1 - \left(\boldsymbol{O}_{m}^{3}\right)^{2}\right) \cdot (1) \quad . \tag{4.1.2.18}$$

The results obtained by moving to the second and first layer are simply equivalent to those for the third with the addition of relevant terms in the decomposition of the partial derivatives. For the sake of brevity, the results for each of the layers, for a single output neuron (***k***) and single training observation, are listed in Table 4-2.

Table 4-2. Jacobian calculations for each layer of isolated IANN

Layer	$\frac{\partial \hat{x}_k}{\partial w_{i,m}^L}$	$\frac{\partial \hat{x}_k}{\partial b_m^L}$
1—Mapping Layer	$\left(w_{m,k}^2\right)\cdot\left(1-\left(O_m^1\right)^2\right)\cdot\left(\frac{\partial \hat{x}_k}{\partial w_{i,m}^2}\right)\cdot \bar{x}$	$\left(w_{m,k}^2\right)\cdot\left(1-\left(O_m^1\right)^2\right)\cdot\left(\frac{\partial \hat{x}_k}{\partial w_{i,m}^2}\right)$
2—Bottleneck Layer	$\left(w_{m,k}^3\right)\cdot\left(1-\left(O_m^2\right)^2\right)\cdot\left(\frac{\partial \hat{x}_k}{\partial w_{i,m}^3}\right)\cdot\left(O_i^1\right)$	$\left(w_{m,k}^3\right)\cdot\left(1-\left(O_m^2\right)^2\right)\cdot\left(\frac{\partial \hat{x}_k}{\partial w_{i,m}^3}\right)$
3—Demapping Layer	$\left(w_{m,k}^4\right)\cdot\left(1-\left(O_m^3\right)^2\right)\cdot\left(O_i^2\right)$	$\left(w_{m,k}^4\right)\cdot\left(1-\left(O_m^3\right)^2\right)\cdot\left(\frac{\partial \hat{x}_k}{\partial w_{i,m}^4}\right)$
4—Output Layer	O_I^3	1

These formulas are used to calculate the Jacobian for each of the process variables. For example, if an AANN uses p variables to predict the corrected $\boldsymbol{p}$ variables, then there would be a total of $\boldsymbol{p}$ individual Jacobian matrices of dimension $\boldsymbol{n_{trn}} \times \boldsymbol{P}$.

For the calculation of Jacobian matrix (**f**), the above sequence of calculations is used for the new query observation only, while the calculation of the matrix of partial derivatives (**F**) is accomplished by running each of the training observations through the sequence. Therefore, the dimensions of **f** are $\boldsymbol{n_{trn}} \times 1$.

The resulting matrices may then be used in the below equation to determine the prediction intervals for each of the output variables of the AANN.

$$\hat{\boldsymbol{y}} \pm \boldsymbol{t}_{n_{trn}-p,\alpha/2}\sqrt{\hat{\sigma}_\varepsilon^2 + \hat{\sigma}_\varepsilon^2\ \mathbf{f}^T\left[\mathbf{F}^T\mathbf{F}\right]^{-1}\mathbf{f} + \boldsymbol{Bias}^2}\ . \tag{4.1.2.19}$$

Additionally, Eq. (4.1.2.19) may be simplified by assuming that a sufficient number of training observations have been used to train the AANN, such that the asymptotic $\boldsymbol{t}$-value of 2 for a 95% confidence level may be used. This approximation results in the final equation for the prediction interval of an AANN:

$$\hat{\boldsymbol{y}} \pm 2\sqrt{\hat{\sigma}_\varepsilon^2 + \hat{\sigma}_\varepsilon^2\ \mathbf{f}^T\left[\mathbf{F}^T\mathbf{F}\right]^{-1}\mathbf{f} + \boldsymbol{Bias}^2}\ . \tag{4.1.2.20}$$

Again, the confidence interval can be constructed from the prediction interval by neglecting the stand-alone $\hat{\sigma}_\varepsilon^2$ term beneath the square root:

$$\boldsymbol{E}(\hat{\boldsymbol{y}}) \pm 2\sqrt{\hat{\sigma}_\varepsilon^2\ \mathbf{f}^T\left[\mathbf{F}^T\mathbf{F}\right]^{-1}\mathbf{f} + \boldsymbol{Bias}^2}\ . \tag{4.1.2.21}$$

4.2 Memory-Based Modeling Techniques

To derive the uncertainty equations for AAKR and AAMSET, the analytic equations for estimating model variance must be developed. Recall that AAKR and AAMSET perform a weighted average on a set of memory vectors. Therefore, the variance propagation equations for both modeling techniques are very similar to that of a simple weighted average.

To begin, consider the following equation for a weighted sum, where w_i are the weights:

$$\mathbf{Y} = \sum_{i=1}^{n} w_i \, \mathbf{X}_i \quad . \tag{4.2.1}$$

The expected value and variance of the weighted sum are given by the following two equations [Morgan and Henrion 1990]:

$$E(\mathbf{Y}) = \sum_{i=1}^{n} w_i \, \mathbf{X}_i \quad , \tag{4.2.2}$$

$$Var(\mathbf{Y}) = \sum_{i=1}^{n} w_i^2 \; Var(\mathbf{X}_i) + 2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_i \, w_j \; Cov(\mathbf{X}_i, \mathbf{X}_j) \quad . \tag{4.2.3}$$

If the values of **X** are assumed to be independent, the equation for variance becomes

$$Var(\mathbf{Y}) \approx \sum_{i=1}^{n} w_i^2 \; Var(\mathbf{X}_i) \quad . \tag{4.2.4}$$

This methodology may now be applied to a weighted average. Using the same notation as the weighted sum, the weighted average of **X** may be written as

$$\mathbf{Y} = \frac{\sum_{i=1}^{n} (w_i \, \mathbf{X}_i)}{\sum_{i=1}^{n} w_i} \quad . \tag{4.2.5}$$

Because the sum of the weights is a constant for a single weighted average, the variance is estimated by

$$Var(\mathbf{Y}) = \frac{\sum_{i=1}^{n} w_i^2 \; Var(\mathbf{X}_i)}{\sum_{i=1}^{n} w_i^2} \quad . \tag{4.2.6}$$

This equation will now be adapted to construct the uncertainty intervals for AAKR and AAMSET.

Recall that the fundamental operation used by AAKR to estimate parameters is the weighted average.

$$\hat{\mathbf{x}} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{X}}{\boldsymbol{a}} \quad , \tag{4.2.7}$$

where

$\hat{\mathbf{x}}$ is a $1 \times \boldsymbol{p}$ matrix of estimates of $\mathbf{x}$,
$\mathbf{w}$ is a $1 \times \boldsymbol{n_m}$ matrix of weights,
X is a $\boldsymbol{n_m} \times \boldsymbol{p}$ matrix of memory vectors,
$\boldsymbol{a} = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n_m}} \boldsymbol{w_i}$ where $\boldsymbol{w_i}$ is the weight of the $\boldsymbol{i}$th memory vector.

For AAKR, $\mathbf{w}$ is determined by evaluating the Gaussian kernel of bandwidth $\boldsymbol{h}$ with the distance $\mathbf{d}$.

$$\mathbf{w} = \boldsymbol{K}(\mathbf{d}, \boldsymbol{h}) = \frac{1}{\sqrt{2\pi\,\boldsymbol{h}^2}} \boldsymbol{e}^{-\mathbf{d}^2/\boldsymbol{h}^2} \quad . \tag{4.2.8}$$

Using these definitions, the variance of the uncertainty for AAKR is found to be the following, where the noise variance is denoted by $\hat{\boldsymbol{\sigma}}_{\varepsilon}^2$:

$$\boldsymbol{Var}(\hat{\mathbf{x}}) = \frac{\sum_{\boldsymbol{i}=1}^{\boldsymbol{n_m}} \boldsymbol{w_i}^2 \; \hat{\sigma}_{\varepsilon}^2}{\sum_{\boldsymbol{i}=1}^{\boldsymbol{n_m}} \boldsymbol{w_i}^2} \quad . \tag{4.2.9}$$

The above equation may now be rewritten in a more compact form in terms of the hat matrix $\mathbf{h}$:

$$\mathbf{h}^{\mathrm{T}} = \left[\frac{\boldsymbol{w}_1}{\boldsymbol{a}} \quad \frac{\boldsymbol{w}_2}{\boldsymbol{a}} \quad \cdots \quad \frac{\boldsymbol{w}_{\boldsymbol{n_m}}}{\boldsymbol{a}} \right] \quad . \tag{4.2.10}$$

where

$\mathbf{h}^{\mathrm{T}} = \left[\frac{\boldsymbol{w}_1}{\boldsymbol{a}} \quad \frac{\boldsymbol{w}_2}{\boldsymbol{a}} \quad \cdots \quad \frac{\boldsymbol{w}_{\boldsymbol{n_m}}}{\boldsymbol{a}} \right]$ is the hat matrix,

$\hat{\boldsymbol{\sigma}}_{\varepsilon}^2$ is a $1 \times \boldsymbol{p}$ matrix of variance estimates of the noise on $\mathbf{x}$.

Alternatively, in AAMSET, $\mathbf{w}$ is determined by evaluating:

$$\mathbf{w} = \mathbf{x} \otimes \mathbf{X}^{\mathrm{T}} \quad . \tag{4.2.11}$$

Unlike AAKR, the AAMSET weights are normalized by multiplying $\mathbf{w}^{\mathrm{T}}$ by the inverse of the normalizing matrix $\mathbf{N}$. Therefore, the hat matrix, $\mathbf{h}$, for AAMSET is given by:

$$\mathbf{h}^{\mathrm{T}} = \mathbf{w}^{\mathrm{T}}\mathbf{N}^{-1} \quad . \tag{4.2.12}$$

Finally, this definition for **h** may be used in Eq. (4.2.10) to estimate the prediction variance of an AAMSET model.

The variance estimates for AAKR and AAMSET may now be incorporated into the general prediction interval equation.

$$\hat{\mathbf{x}} \pm \boldsymbol{t}_{n_m - p, \alpha/2} \sqrt{\hat{\sigma}_\varepsilon^2 + \mathbf{h}^{\mathrm{T}}\mathbf{h}\, \hat{\sigma}_\varepsilon^2 + \boldsymbol{Bias}^2} \quad . \tag{4.2.13}$$

Additionally, the above equation may be simplified by assuming that a sufficient number of memory vectors have been used in the AAKR and AAMSET models, such that the asymptotic ***t***-value of 2 for a 95% confidence level may be used. This approximation results in the final equation for constructing the prediction intervals for AAKR and AAMSET models:

$$\hat{\mathbf{x}} \pm 2 \sqrt{\hat{\sigma}_\varepsilon^2 + \mathbf{h}^{\mathrm{T}}\mathbf{h}\, \hat{\sigma}_\varepsilon^2 + \boldsymbol{Bias}^2} \quad . \tag{4.2.14}$$

The confidence interval for AAKR and AAMSET models is then the following:

$$\boldsymbol{E}(\hat{\mathbf{x}}) \pm 2 \sqrt{\mathbf{h}^{\mathrm{T}}\mathbf{h}\, \hat{\sigma}_\varepsilon^2 + \boldsymbol{Bias}^2} \quad . \tag{4.2.15}$$

4.3 Analytic Summary

In this section, the analytic equations for determining the uncertainty of the AANN, AAKR, and AAMSET models were derived. From these equations, prediction and confidence intervals can be constructed for each modeling technique. Although these analytic equations are complex and have many assumptions, they are generally regarded to give a robust estimate of model uncertainty. In the next section, Monte Carlo-based uncertainty analyses are presented as an alternative to the analytic methods. In comparison to the analytic analyses, the Monte Carlo techniques are more straightforward, have fewer assumptions, and are generally easier to understand. Still, due to the computational intensity of the Monte Carlo techniques, the analytic methods are often the only practical means of quantifying a model's uncertainty. In subsequent applications it is shown that the analytical and Monte Carlo uncertainty prediction techniques produce similar results.

5. MONTE CARLO METHODS

Monte Carlo analysis gets its name from the city/state of Monte Carlo, Monaco, which is famous for its casinos. The games in the casinos, such as roulette, slot machines, and dice, all exhibit random behavior. The random behavior in these games is similar to how Monte Carlo methods select the variable values used to construct a model. In general, Monte Carlo methods can be described as simulation methods that use sequences of random numbers to perform statistical simulations. In Monte Carlo analysis, rather than making a single prediction based on the "best guess" for each uncertain variable, multiple predictions are made using different values for the uncertain variable, where each possible value for each uncertain variable (one that has a range of possible values) is defined with a probability distribution. The uncertain variables in OLM applications are the training data. The values are uncertain due to the irreducible error that is modeled by a distribution. The basic method is to estimate the noise-free variables and their associated noise distributions. Samples (training data) are then selected, and models are constructed. The uncertainty in the model predictions are determined by characterizing the variations in the predictions caused by the random nature of the training data. In general, Monte Carlo methods provide reasonably accurate descriptions of possible model outcomes. Monte Carlo analyses are often used in the nuclear industry [Rasmussen 2002].

Monte Carlo-based uncertainty estimation methods applicable to OLM have two primary sampling frameworks: one is based upon a wavelet denoising with Latin hypercube sampling (LHS), developed at Argonne National Laboratory [Zavaljevski et al. 2004], and the other is a direct bootstrap, developed at the University of Tennessee [Rasmussen 2003].

The general process used by both Monte Carlo uncertainty estimation algorithms may be described as follows: (1) create a training set by sampling from the original data (algorithm dependent); (2) build a prototype model using the training set; (3) simulate the model with an independent test or validation set; (4) store the parameter estimates in simulation memory; and (5) repeat the process, starting at (1), until the specified number of iterations have run. Following this iterative cycle, the simulation memory is used to develop an estimate of the model's uncertainty by evaluating the prediction variance and estimating the model's bias according to the methods described in Chap. 4. Figure 5-1 depicts this process.

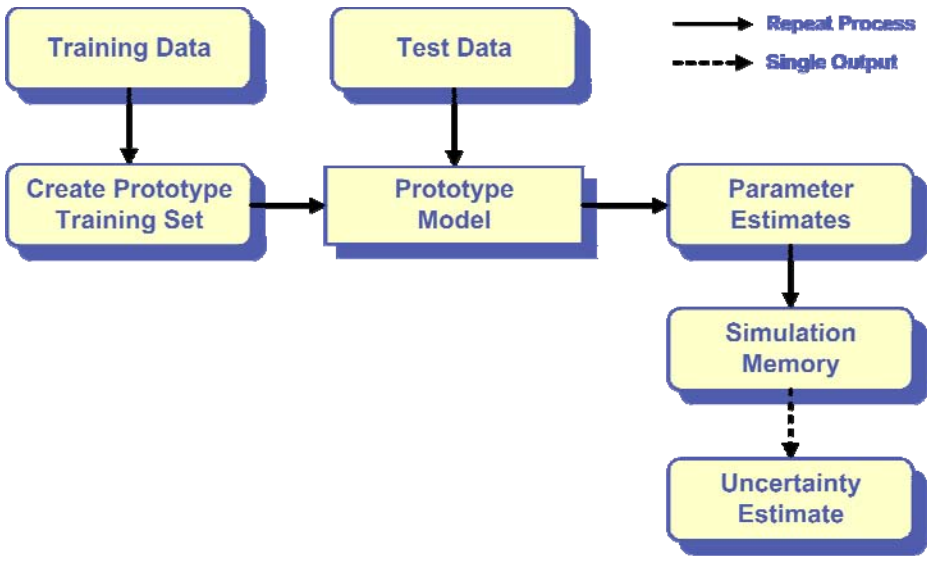


Fig. 5-1. Process diagram for Monte Carlo uncertainty estimation.

5.1 Direct Bootstrap Sampling

The direct bootstrap sampling algorithm is a simplified variation on traditional Monte Carlo methods in that it iteratively samples and tests a set of data, as is shown in Fig. 5-2. The simplification lies in the method of sampling, which is performed by randomly selecting observations directly from the training data with replacement (i.e., an observation may be selected more than once), rather than sampling data with noise simulated from a probability distribution as in most Monte Carlo techniques [Efron and Tibshirani 1993]. From each resultant sampled data set, a new model is built. From this collection of models, the model variation and the average model bias are estimated. Although this approach has a simple architecture, it has been shown by Tibshirani [1996] that it performs comparable to asymptotic uncertainty estimation techniques for neural networks.

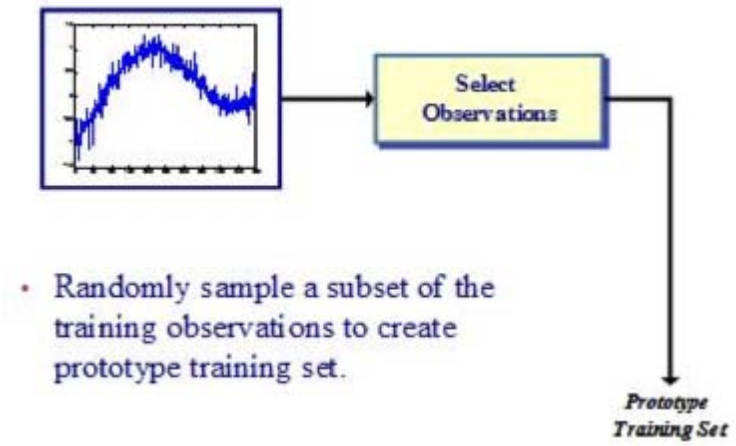


Fig. 5-2. Process diagram for bootstrap uncertainty estimation.

5.2 Latin Hypercube Sampling

The Latin hypercube sampling (LHS) is a method used to reduce the computational burden of conventional MC techniques such as the bootstrap. Rather than randomly sampling from a distribution, LHS techniques sample in a "smart" way to cover the distribution with a fewer number of samples.

MIron [2001] applied wavelet filtering to estimate the "true" process values. A diagram of this process is presented in Fig. 5-3. The "true" values are then subtracted from the data, providing an estimate of the noise which is then modeled with a distribution. This noise distribution is then sampled via LHS to construct prototype training sets [Zavaljevski et al. 2004]. From this preliminary description, it can be seen that this sampling technique is fundamentally composed of two processes: estimating and sampling a noise distribution.

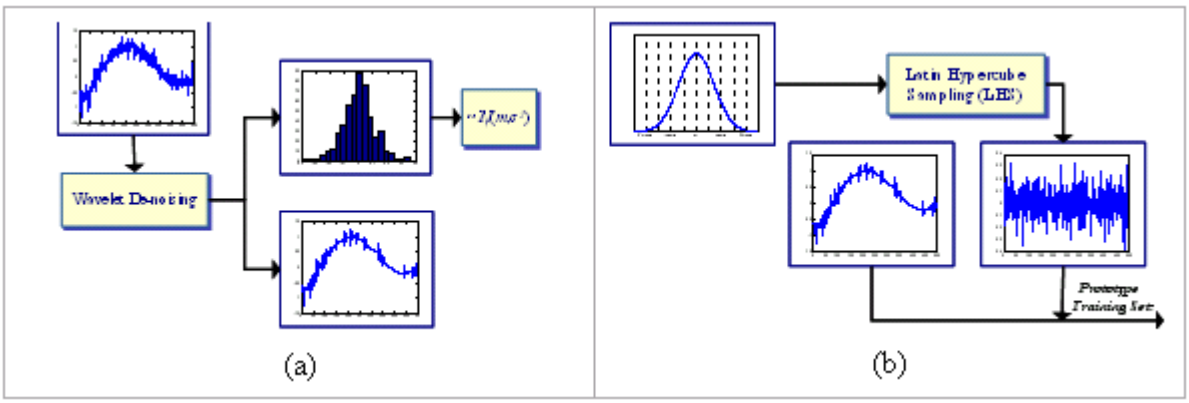


Fig. 5-3. Process diagram LHS with wavelet denoising.

Because the focus of this section is to discuss LHS methodology, the details of wavelet denoising will not be discussed at this time, and the reader is referred to Sect. 3.4.2 of this document, where it is discussed in more detail.

LHS is a stratified sampling technique that is used to sample a distribution evenly [Helton 2004]. This is accomplished by segmenting the distribution into a number of nonoverlapping intervals, commonly termed "bins," with each bin having an equal probability. Each probability "bin" has a restricted range of possible distribution values [Wyss and Jorgensen 1998]. Figure 5-4 shows a normal distribution divided into 10 probability bins, with each bin having a probability of 0.1. Thus, when LHS sampling is used, random values are selected from each bin with equal frequency. Typically, the distribution is segmented into N bins, each having a probability of $1/N$, where N is the number of observations in the training data.

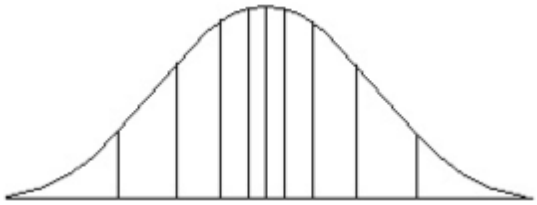


Fig. 5-4. Normal distribution segmented for LHS sampling.

As seen in Fig. 5-5, the result of the LHS is that the noise distribution is sampled evenly, whereas traditional or random methods result in a sampling that closely mimics the original distribution. Thus, in the random methods, the tails of the distribution are not factored in to the final result as they should be. This fact is especially true for highly skewed distributions, or ones with long tails. When using basic sampling methods, it is necessary to conduct many trials to ensure that the unlikely extremities, which have a low probability of occurring and are at the tails of the distribution, are taken into account. Because the full range of the distribution is sampled more evenly and consistently with LHS, it requires fewer trials than the traditional Monte Carlo and bootstrap methods for similar accuracy.

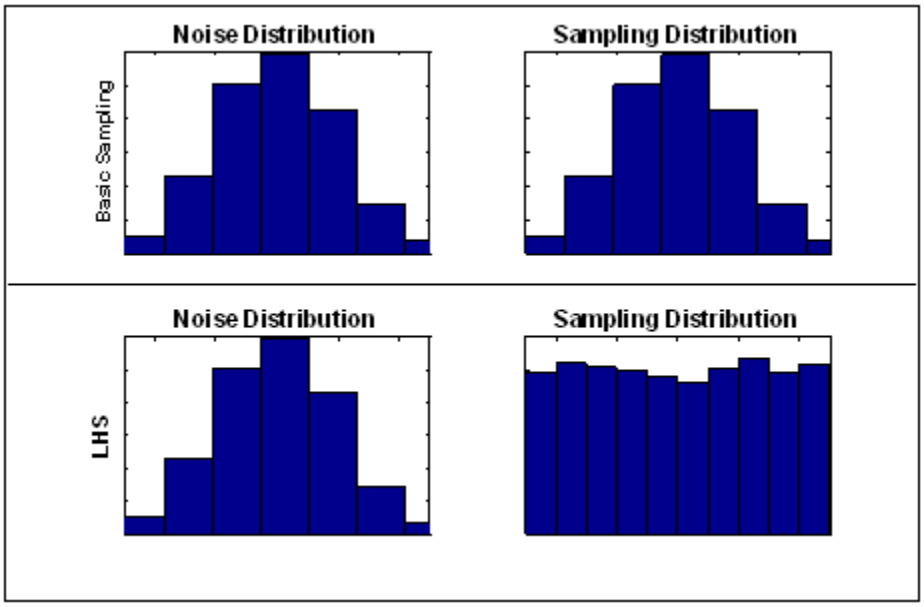


Fig. 5-5. Comparison of basic distribution sampling and LHS.

Although LHS may converge to the true uncertainty value more quickly, it does have its drawbacks. First, the distribution of the data has to be accurately quantified. Miron [2001] addresses the issues and presents several statistical tests for determining the distribution of a given data set. Additionally, extra memory is required because of the stratification that must be retained throughout the sampling.

5.3 Comparison of Latin Hypercube Sampling and Direct Bootstrap Sampling

In this section, variance predictions are determined using both the bootstrap and LHS. The goal is to determine how many runs are needed to provide a stable estimate of a model's variance. This is performed by plotting the model's predicted variance against the number of runs. The model variance (i.e., the variability between the many models constructed with the data and not the noise in the data) is the parameter of concern.

The variance was calculated for an incrementally increasing number of bootstrap replications. Figure 5-6 presents the simulated nuclear power plant data that is to be modeled. In this example, Channel 1 contains the highest contribution of independent noise while Channel 3 has the least. The data set contains 600 observations.

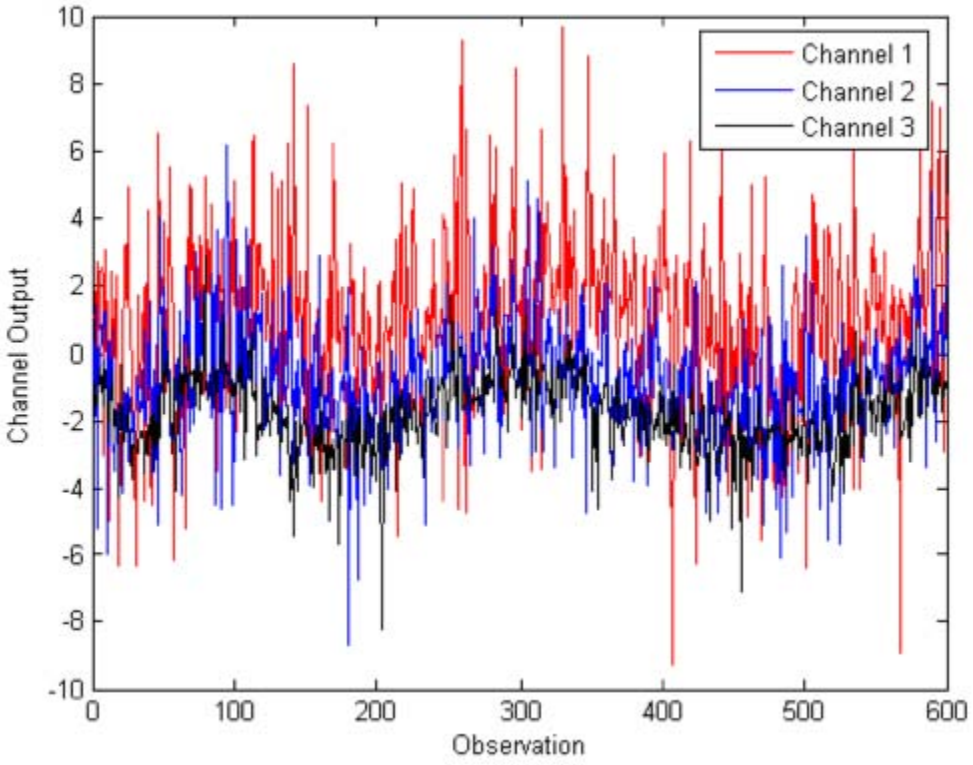


Fig. 5-6. Simulated channel data containing both independent and common noise.

Bootstrap simulations were carried out using these data with varied bootstrap numbers, ranging from n = 10 to 1000, with a step size of 10. Rather than keeping the previous runs and simply appending additional ones on to them to calculate the statistic for a larger number of samples, an entirely new set of bootstrap runs was performed to provide independent samples. Figure 5-7 shows the results. Each of the n bootstrap runs results in a new model, which is used to make predictions on a test data set. The model variance is then calculated for each test observation, which is then averaged to obtain a single variance value for each channel.

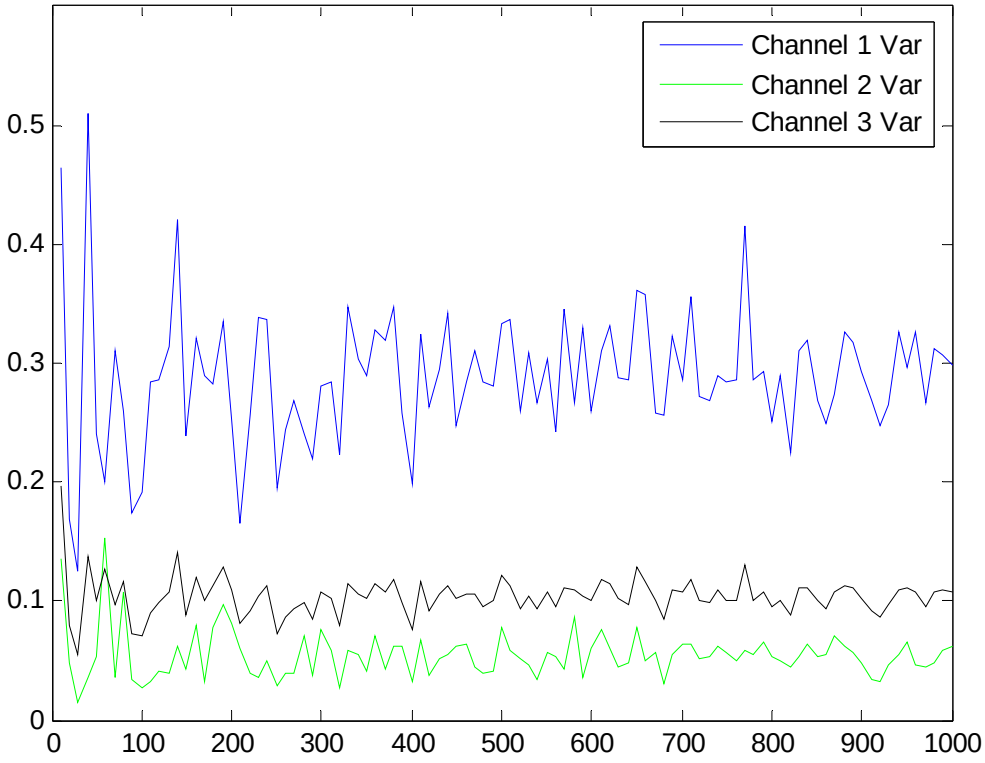


Fig. 5-7. Variance vs number of bootstrap runs.

Figure 5-7 indicates that the bootstrap variance prediction is somewhat unstable and probably unrepeatable with fewer than 300 simulations. Even with 1000 runs, the results have not entirely leveled off. Thus, the acceptability of the bootstrap prediction at this point depends on the level of confidence and accuracy that the estimate must have, which is dictated by its specific application. The figure also shows that, as expected, the noisier channel (Channel 1) requires more bootstrap runs before the estimate of the variance stabilizes.

An identical study was repeated using LHS rather than sampling the data directly using bootstrap. Using LHS requires denoising the signal and modeling the noise distribution so that it can be sampled. For this example, the noise probability distribution was modeled as a normal distribution. The result of this example is shown in Fig. 5-8.

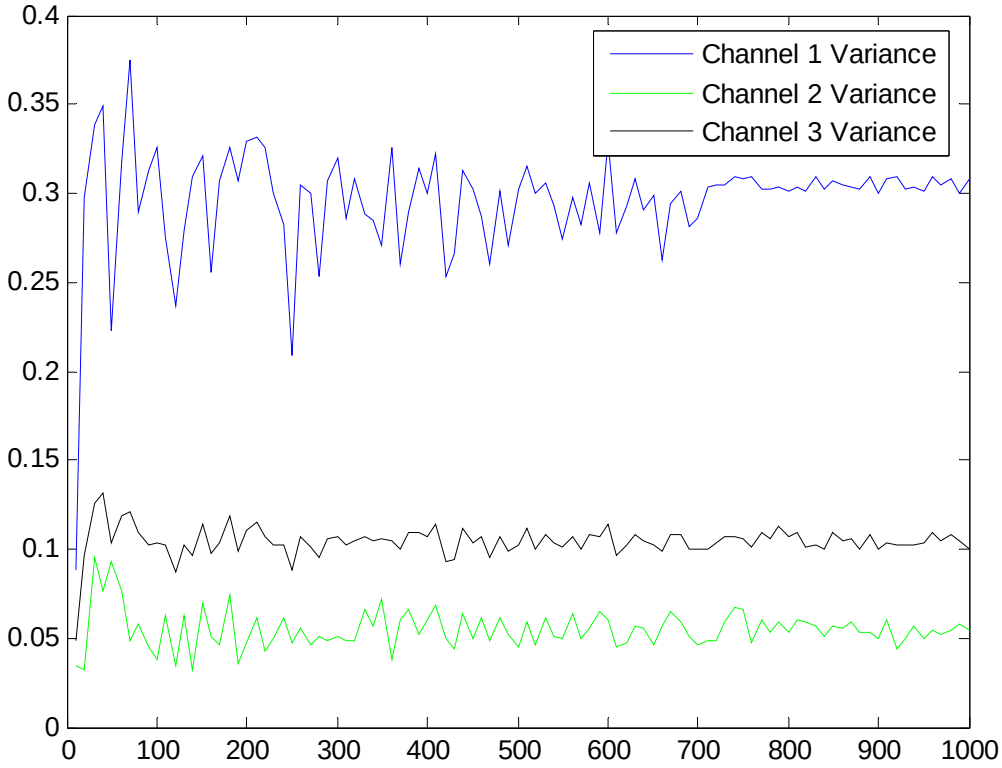


Fig. 5-8. Variance vs number of LHS runs.

As expected, LHS reduced the number of runs necessary to stabilize the Monte Carlo simulation. When the noise distribution is easily determined, LHS proves to be quick, simple, and easy to implement. However, the computation effort and oversight necessary for modeling the noise distributions may result in LHS being more time-consuming; this is especially true if the noise distribution does not fit a normal distribution. In comparison, the bootstrap method is always relatively straightforward. Thus, for many OLM uncertainty analyses, the bootstrap direct sampling method may be preferred over LHS.

5.4 Equations for Uncertainty

To this point, the discussion has focused on how historical data are sampled with only a brief discussion of variance computation. This section will present equations used to combine the Monte Carlo samples into an estimate of model uncertainty.

Suppose a set of training data (X) exists, with $\boldsymbol{n_{trn}}$ observations of $\boldsymbol{p}$ variables, where $\boldsymbol{X_{i,j}}$ is the ith training observation of variable j:

$$\mathbf{X} = \begin{bmatrix} \boldsymbol{X}_{1,1} & \boldsymbol{X}_{1,2} & \cdots & \boldsymbol{X}_{1,\boldsymbol{p}} \\ \boldsymbol{X}_{2,1} & \boldsymbol{X}_{2,2} & \cdots & \boldsymbol{X}_{2,\boldsymbol{p}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{X}_{\boldsymbol{n}_{trn},1} & \boldsymbol{X}_{\boldsymbol{n}_{trn},2} & \cdots & \boldsymbol{X}_{\boldsymbol{n}_{trn},\boldsymbol{p}} \end{bmatrix} . \qquad (5.4.1)$$

To construct a prototype model, n_{trn} samples are selected from **X** with replacement, where the samples are either used to train an AANN or are used to select exemplar vectors for the memory matrix of an AAKR or AAMSET model. For the direct bootstrap technique, the sample would be n_{trn} observations that were randomly selected from **X**. For LHS, the sample would be the selected instances of the sum of the "true" values of **X** and an LHS sample of the noise. $\mathbf{X}^*$ is defined as the Monte Carlo sample of **X** used to create a prototype model and may be written as

$$\mathbf{X}^* = \begin{bmatrix} X^*_{1,1} & X^*_{1,2} & \cdots & X^*_{1,p} \\ X^*_{2,1} & X^*_{2,2} & \cdots & X^*_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ X^*_{n_{trn},1} & X^*_{n_{trn},2} & \cdots & X^*_{n_{trn},p} \end{bmatrix}. \tag{5.4.2}$$

Recall that for the Monte Carlo based uncertainty estimation methods, the prototype models are tested with a second independent data set. If this test set contains $\boldsymbol{n_q}$ observations, it may be written as

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{x}_{1,1} & \boldsymbol{x}_{1,2} & \cdots & \boldsymbol{x}_{1,\boldsymbol{p}} \\ \boldsymbol{x}_{2,1} & \boldsymbol{x}_{2,2} & \cdots & \boldsymbol{x}_{2,\boldsymbol{p}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_{\boldsymbol{n_q},1} & \boldsymbol{x}_{\boldsymbol{n_q},2} & \cdots & \boldsymbol{x}_{\boldsymbol{n_q},\boldsymbol{p}} \end{bmatrix}. \tag{5.4.3}$$

For an auto-associative empirical model, the predictions for a prototype model are estimates of x and therefore are written as

$$\hat{\mathbf{x}}^* = \begin{bmatrix} \hat{\boldsymbol{x}}^*_{1,1} & \hat{\boldsymbol{x}}^*_{1,2} & \cdots & \hat{\boldsymbol{x}}^*_{1,\boldsymbol{p}} \\ \hat{\boldsymbol{x}}^*_{2,1} & \hat{\boldsymbol{x}}^*_{2,2} & \cdots & \hat{\boldsymbol{x}}^*_{2,\boldsymbol{p}} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\boldsymbol{x}}^*_{\boldsymbol{n_q},1} & \hat{\boldsymbol{x}}^*_{\boldsymbol{n_q},2} & \cdots & \hat{\boldsymbol{x}}^*_{\boldsymbol{n_q},\boldsymbol{p}} \end{bmatrix}. \tag{5.4.4}$$

The number of prototype models created and tested is defined as N, so there will be N versions of $\hat{\mathbf{x}}^*$. To distinguish between these values, $\hat{\mathbf{x}}^{\boldsymbol{k}*}$ is defined as the prediction of the kth prototype model for **x**, where $\hat{\boldsymbol{x}}^{\boldsymbol{k}*}_{\boldsymbol{i},\boldsymbol{j}}$ is the ith observation of the jth variable of $\hat{\mathbf{x}}^{\boldsymbol{k}*}$.

The expected value for the prediction of the ith observation of the jth variable is simply defined as the average of the $\boldsymbol{N}$ model predictions.

$$\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{i},\boldsymbol{j}}\right) = \frac{\sum_{\boldsymbol{k}=1}^{\boldsymbol{N}} \hat{\boldsymbol{x}}^{\boldsymbol{k}*}_{\boldsymbol{i},\boldsymbol{j}}}{\boldsymbol{N}}. \tag{5.4.5}$$

Therefore, the expected prediction of $\boldsymbol{x}$ can be written as

$$\boldsymbol{E}\left(\hat{\mathbf{x}}\right)=\frac{1}{\boldsymbol{N}}\cdot\begin{bmatrix}\sum_{k=1}^{N}\hat{\boldsymbol{x}}_{1,1}^{k*} & \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{1,2}^{k*} & \cdots & \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{1,p}^{k*}\\ \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{2,1}^{k*} & \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{2,2}^{k*} & \cdots & \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{2,p}^{k*}\\ \vdots & \vdots & \ddots & \vdots\\ \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{n_q},1}^{k*} & \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{n_q},2}^{k*} & \cdots & \sum_{k=1}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{n_q},p}^{k*}\end{bmatrix} \quad . \tag{5.4.6}$$

$$\boldsymbol{E}\left(\hat{\mathbf{x}}\right)=\begin{bmatrix}\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{1,1}\right) & \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{1,2}\right) & \cdots & \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{1,p}\right)\\ \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{2,1}\right) & \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{2,2}\right) & \cdots & \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{2,p}\right)\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},1}\right) & \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},2}\right) & \cdots & \boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},p}\right)\end{bmatrix}$$

In a similar fashion, the variance of the *i*th observation of the *j*th variable is defined as

$$\boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{\boldsymbol{i},\boldsymbol{j}}\right)=\frac{\sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{\boldsymbol{i},\boldsymbol{j}}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{i},\boldsymbol{j}}\right)\right]^{2}}{\boldsymbol{N}-1} \quad . \tag{5.4.7}$$

Therefore, the variance of the model predictions can be written as

$$\boldsymbol{Var}\left(\hat{\mathbf{x}}\right)=\frac{1}{\boldsymbol{N}-1}\cdot\begin{bmatrix}\sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{1,1}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{1,1}\right)\right]^{2} & \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{1,2}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{1,2}\right)\right]^{2} & \cdots & \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{1,p}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{1,p}\right)\right]^{2}\\ \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{2,1}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{2,1}\right)\right]^{2} & \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{2,2}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{2,2}\right)\right]^{2} & \cdots & \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{2,p}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{2,p}\right)\right]^{2}\\ \vdots & \vdots & \ddots & \vdots\\ \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{\boldsymbol{n_q},1}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},1}\right)\right]^{2} & \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{\boldsymbol{n_q},2}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},2}\right)\right]^{2} & \cdots & \sum_{k=1}^{N}\left[\hat{\boldsymbol{x}}_{\boldsymbol{n_q},p}^{k*}-\boldsymbol{E}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},p}\right)\right]^{2}\end{bmatrix} \quad . \tag{5.4.8}$$

$$\boldsymbol{Var}\left(\hat{\mathbf{x}}\right)=\begin{bmatrix}\boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{1,1}\right) & \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{1,2}\right) & \cdots & \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{1,p}\right)\\ \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{2,1}\right) & \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{2,2}\right) & \cdots & \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{2,p}\right)\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},1}\right) & \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},2}\right) & \cdots & \boldsymbol{Var}\left(\hat{\boldsymbol{x}}_{\boldsymbol{n_q},p}\right)\end{bmatrix}$$

The variance for each of the $\boldsymbol{p}$ variables is conservatively selected to be the 95th percentile value. This value is calculated by arranging the $\boldsymbol{n_q}$ variance estimates in ascending order and selecting the 95th

percentile largest value. This variance estimate is now incorporated into the general prediction interval equation [Eq. (4.2.2.6)] as follows:

$$\hat{\mathbf{x}} \pm \boldsymbol{t}_{n_q - p, \alpha/2} \sqrt{\hat{\sigma}_{\varepsilon}^2 + \boldsymbol{Var}\left(\hat{\mathbf{x}}\right) + \boldsymbol{Bias}^2} \quad . \tag{5.4.9}$$

Here, $\hat{\sigma}_{\varepsilon}^2$ is an estimate of the noise variance and the bias is derived according to methods described earlier.

Because the uncertainty found by the Monte Carlo algorithms requires many iterations, it is not practical for near-real-time use. Therefore, a single point uncertainty is calculated from the Monte Carlo estimate for a set of unfaulted test data. This single point estimate is generally defined to be the 95th percentile Monte Carlo uncertainty estimate for the test data. Furthermore, since Monte Carlo techniques require the construction of many prototype models, the large computational burdens of neural network training generally restrict the use of AANNs to applications where small network architectures may be used.

5.5 Monte Carlo Summary

In this chapter, the general process used by Monte Carlo uncertainty estimation methods was explained. The chapter also examined two Monte Carlo techniques commonly used in OLM: the direct bootstrap and the LHS method. Overall, Monte Carlo techniques are more intuitive than the analytic methods and are also generally regarded as more accurate [Rasmussen 2003]. However, the accuracy of these methods depends upon the number of simulations performed. This requirement of a large number of simulations, and the resulting computational burden, could prohibit the use of Monte Carlo techniques in certain OLM applications. The following section compares the analytic and Monte Carlo techniques using both simulated and actual nuclear plant data.

6. COMPARISON OF UNCERTAINTY ESTIMATION METHODS

This chapter will use both artificially generated and actual nuclear plant data to compare the uncertainty estimation methods described in Chaps. 4 and 5. First, the methods used to estimate the noise variance and model bias will be discussed. Next, the analytic and Monte Carlo methodologies will be applied to the artificially generated and actual plant data. Finally, the results will be synthesized to provide a holistic perspective of each method's advantages and drawbacks, as well as providing guidelines for proper implementation.

Recall that the general equation for the confidence interval (CI) of an auto-associative model (i.e., $\boldsymbol{x}$ is estimated by $\hat{\boldsymbol{x}}$) for a 95% confidence level is given by:

$$\hat{\boldsymbol{f}}(\mathbf{x}) \pm 2\sqrt{\boldsymbol{Var}(\hat{\mathbf{x}}) + \boldsymbol{Bias}^2}\ , \tag{6.1}$$

where

$\hat{\boldsymbol{f}}(\boldsymbol{x})$ is an estimate of the true values of $\boldsymbol{x}$,

$\boldsymbol{Var}(\hat{\boldsymbol{x}})$ is the model prediction variance,

Bias is the model prediction bias.

For a single query observation, the dimensions of $\boldsymbol{x}$, $\hat{\boldsymbol{f}}(\boldsymbol{x})$, $\boldsymbol{Var}(\hat{\boldsymbol{x}})$, and ***Bias*** are $1 \times \boldsymbol{p}$, with $\boldsymbol{p}$ being the number of variables or sensors being modeled. This equation will serve as a foundation for the examples discussed in this chapter.

Next, the CI for the residuals or error is given by Eq. (6.2).

$$\hat{\boldsymbol{f}}(\boldsymbol{\varepsilon}) \pm 2\sqrt{\boldsymbol{Var}(\hat{\boldsymbol{x}}) + \boldsymbol{Bias}^2}\ , \tag{6.2}$$

where

$\hat{\boldsymbol{f}}(\boldsymbol{\varepsilon})$ is an estimate of the true, noise-free residuals with $\boldsymbol{\varepsilon} = \hat{\mathbf{x}} - \mathbf{x}$;

$\boldsymbol{Var}(\hat{\mathbf{x}})$ is the variance of the model's predictions;

Bias is the model's prediction bias.

6.1 Noise Variance Estimation

For this example, the variance of the noise will be estimated by applying the wavelet denoising methodology described in Sect. 4.4 and illustrated by the following equations.

$$\boldsymbol{\varepsilon}_i = \mathbf{X}_i - \hat{\boldsymbol{f}}(\mathbf{X}_i)\,, \tag{6.1.1}$$

where

ε_i is an estimate of the noise on the ith training observation, $\mathbf{X}_i$;

$\hat{f}(\mathbf{X}_i)$ is an estimate of the "true" value for the ith training observation, $\mathbf{X}_i$, found by applying the wavelet denoising algorithm.

For this example, the variance of the noise is assumed to be heteroscadastic, or constant, for the $\boldsymbol{p}$ variables (i.e., $\hat{\sigma}_{\varepsilon}^2$ is an $1 \times \boldsymbol{p}$ matrix of variance estimates):

$$\hat{\sigma}_{\varepsilon}^2 = \frac{1}{\boldsymbol{n}_{trn} - 1} \sum_{i=1}^{\boldsymbol{n}_{trn}} \left[\boldsymbol{\varepsilon}_i - \boldsymbol{E}(\boldsymbol{\varepsilon}) \right]^2 \quad . \tag{6.1.2}$$

where

$\boldsymbol{n}_{trn}$ is the number of training observations;

$\boldsymbol{E}(\boldsymbol{\varepsilon}) = \frac{1}{\boldsymbol{n}_{trn}} \sum_{i=1}^{\boldsymbol{n}_{trn}} \boldsymbol{\varepsilon}_i$ is the estimated noise's expected value, which is most often found to be zero or near zero for drift free data, $\boldsymbol{E}(\boldsymbol{\varepsilon}) \approx 0$;

$\hat{\sigma}_{\varepsilon}^2$ is an estimate of the variance of the noise on $\mathbf{X}$.

6.2 Bias Estimation

For these examples, the model prediction bias is estimated by applying the bias-variance decomposition equation discussed in Sects. 4.2.1 and 4.3.2. The MSE is given by the following equation, where $\mathbf{X}_{tst}$ is an $\boldsymbol{n}_{tst} \times \boldsymbol{p}$ matrix of unfaulted test data and $\boldsymbol{n}_{tst}$ is the number of test observations:

$$\boldsymbol{MSE}\left(\hat{\mathbf{X}}_{tst}\right) = \frac{1}{\boldsymbol{n}_{tst}} \sum_{i=1}^{\boldsymbol{n}_{tst}} \left(\hat{\mathbf{X}}_{tst,i} - \mathbf{X}_{tst,i} \right)^2 \quad , \tag{6.2.1}$$

where $\mathbf{X}_{tst,i}$ and $\hat{\mathbf{X}}_{tst,i}$ are the ith test observation and its estimated value, respectively.

Notice that the MSE has dimensions $1 \times \boldsymbol{p}$. Furthermore, under the assumption of heteroscadastic variance, $\hat{\sigma}_{\varepsilon}^2$ has dimensions $1 \times \boldsymbol{p}$. If the analytic equations are used to estimate the model prediction variance, then the dimensions of the estimated variances have the same dimensions as $\mathbf{X}_{tst}$, specifically $\boldsymbol{n}_{tst} \times \boldsymbol{p}$. An $\boldsymbol{n}_{tst} \times 1$ matrix of ones, $\mathbf{A}$, is introduced at this point to copy the $\boldsymbol{p}$ MSE and error variance values for each of the n_{tst} observations. The model's squared bias, $\left[\boldsymbol{Bias}\left(\hat{\mathbf{X}}_{tst}\right) \right]^2$, is found as follows:

$$\left[\boldsymbol{Bias}\left(\hat{\mathbf{X}}_{tst}\right) \right]^2 = \mathbf{A} \cdot \boldsymbol{MSE}\left(\hat{\mathbf{X}}_{tst}\right) - \boldsymbol{Var}\left(\hat{\mathbf{X}}_{tst}\right) - \mathbf{A} \cdot \hat{\sigma}_{\varepsilon}^2 \quad . \tag{6.2.2}$$

If the model uncertainty is estimated according to a Monte Carlo resampling technique, then the estimated variance is the mean prototype model variance and has dimensions $1 \times \boldsymbol{p}$. In this case, $\mathbf{A}$ must also be multiplied by the variance term.

For this example it is assumed that the model bias is constant for each variable and may be approximated by its expected value. Additionally, if the expected squared bias is found to be negative (i.e., the sum model prediction variance and estimated noise variance is larger than the test MSE), then it is set to zero. These assumptions give rise to the final equation that estimates the model's bias in the form of a $1 \times \boldsymbol{p}$ matrix, $\boldsymbol{Bias}$:

$$\boldsymbol{Bias} = \begin{cases} \sqrt{\boldsymbol{E}\left(\left[\boldsymbol{Bias}\left(\hat{\mathbf{X}}_{tst}\right)\right]^2\right)} & \text{if } \boldsymbol{E}\left(\left[\boldsymbol{Bias}\left(\hat{\mathbf{X}}_{tst}\right)\right]^2\right) \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{6.2.3}$$

where

$\boldsymbol{E}\left(\left[\boldsymbol{Bias}\left(\hat{\mathbf{X}}_{tst}\right)\right]^2\right) = \frac{1}{\boldsymbol{n}_{tst}} \sum_{i=1}^{\boldsymbol{n}_{tst}} \left[\boldsymbol{Bias}\left(\hat{\mathbf{X}}_{i,tst}\right)\right]^2$ is the expected squared model bias,

$\left[\boldsymbol{Bias}\left(\hat{\mathbf{X}}_{i,tst}\right)\right]^2$ is the squared model bias for the ith test prediction.

6.3 Overview of Methodology

In this section, the analysis procedures used in the subsequent examples are presented in detail.

6.3.1 Analytic methodologies

The procedure for applying the analytic uncertainty equations is described by the following process. The first five steps develop the uncertainty estimate, and the final three steps use the uncertainty to analyze the data. The process for calculating the point-wise, analytic uncertainty is presented in Fig. 6-1.

1. Load data, detect and correct outliers, and divide data into training, test, and validation sets. The training set is used to create a model, the test data are used to quantify the bias, and the validation data are used as actual query data.
2. Denoise test the data with the wavelet denoising algorithm to estimate the noise variance.
3. Develop empirical model with the training data.
4. Predict the "correct" sensor values using an empirical model and predict the variance of the estimates with the analytic equations for the test data.
5. Calculate MSE of test predictions. Apply Eq. (6.2.2) with the point-wise variance estimates and noise variances to estimate the model bias. For this analysis a conservative estimate of the bias is calculated to be the 95% percentile of the estimated bias distribution.
6. Predict the "correct" sensor values and the variance of the estimates with the analytic equations for the validation data.
7. Denoise the validation data and evaluate Eq. (6.1) with the model variance and bias to estimate the prediction CIs.
8. Denoise the prediction residuals of the validation data and evaluate Eq. (6.2) with the model variance and bias to estimate the residual CIs.

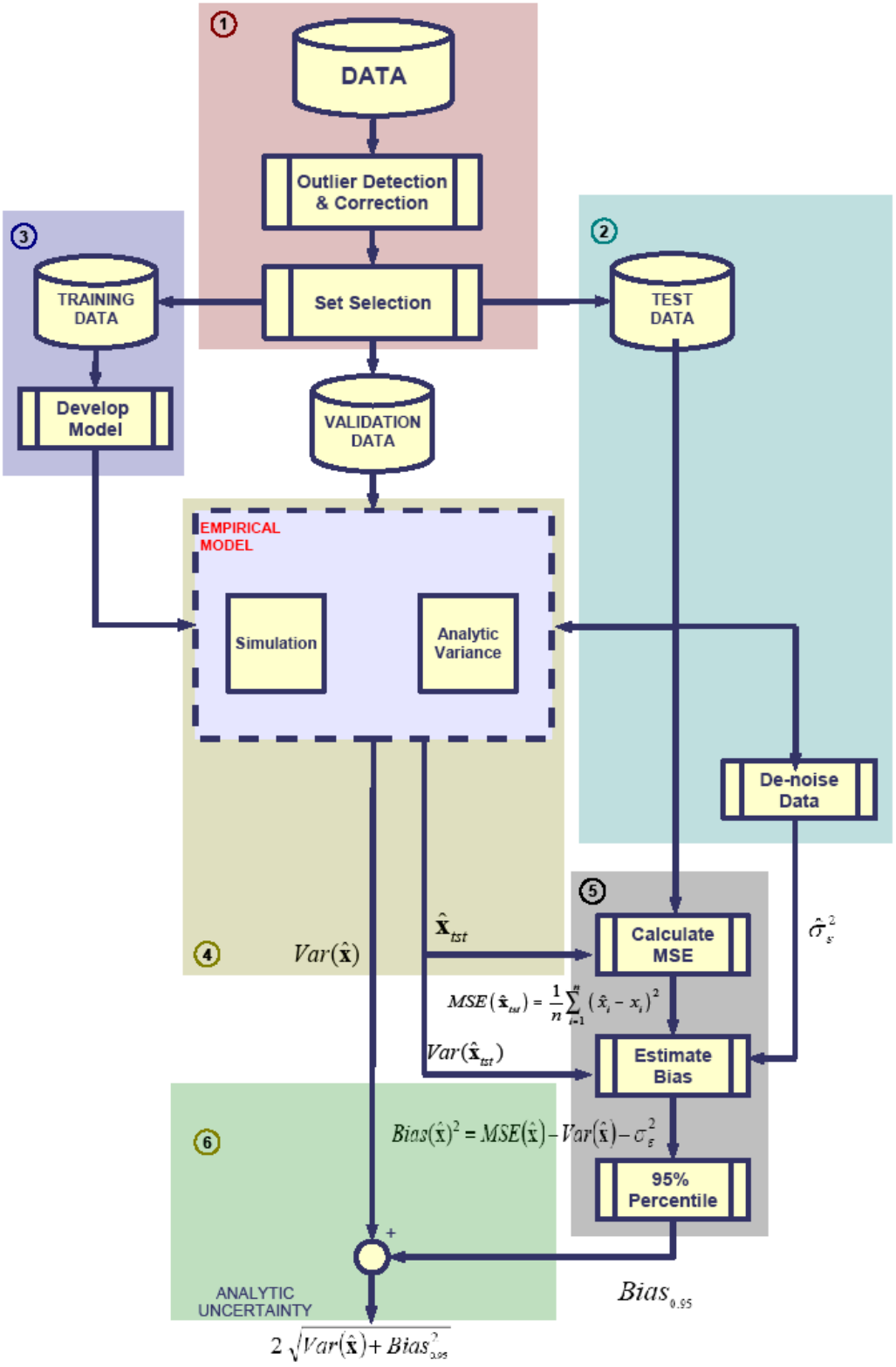


Fig. 6-1. Process diagram for analytic uncertainty estimation of an empirical model.

6.3.2 Monte Carlo methodologies

The procedure for applying the Monte Carlo resampling techniques is described by the following process. The first six steps develop the uncertainty estimate, and the final three use the uncertainty to analyze the data. The methodology for calculating the 95% percentile Monte Carlo uncertainty is presented in Fig. 6-2. Notice that since the Monte Carlo uncertainty estimate is determined entirely from the training and test data, that the validation is not used in the process depicted in Fig, 6-2.

1. Load data, detect and correct outliers, and divide data into training, test, and validation data sets.
2. Denoise the test data with the wavelet denoising algorithm in order to estimate the noise variance.
3. Optimize empirical model architecture with the training data.
4. Predict the expected "correct" sensor values and the variance of the estimates for the test data by resampling the training data, developing and testing multiple prototype models. The single point model variance is estimated by calculating the 95% percentile of the estimated variance distribution.
5. Calculate MSE of expected test predictions. Apply Eq. (6.2.2) with the single point variance estimates and noise variances to estimate the model bias. For this analysis a conservative estimate of the bias is calculated to be the 95% percentile of the estimated bias distribution.
6. Combine the estimate of the model variance and bias to form a 95% percentile uncertainty prediction.
7. Using the empirical models, predict the "correct" sensor values for the validation data.
8. Denoise the validation data and evaluate Eq. (6.1) with the single point uncertainty estimates to form the Monte Carlo prediction CIs.
9. Denoise the residuals for the predictions of the validation data and evaluate Eq. (6.2) with the single point uncertainty estimates to form the Monte Carlo residual CIs.

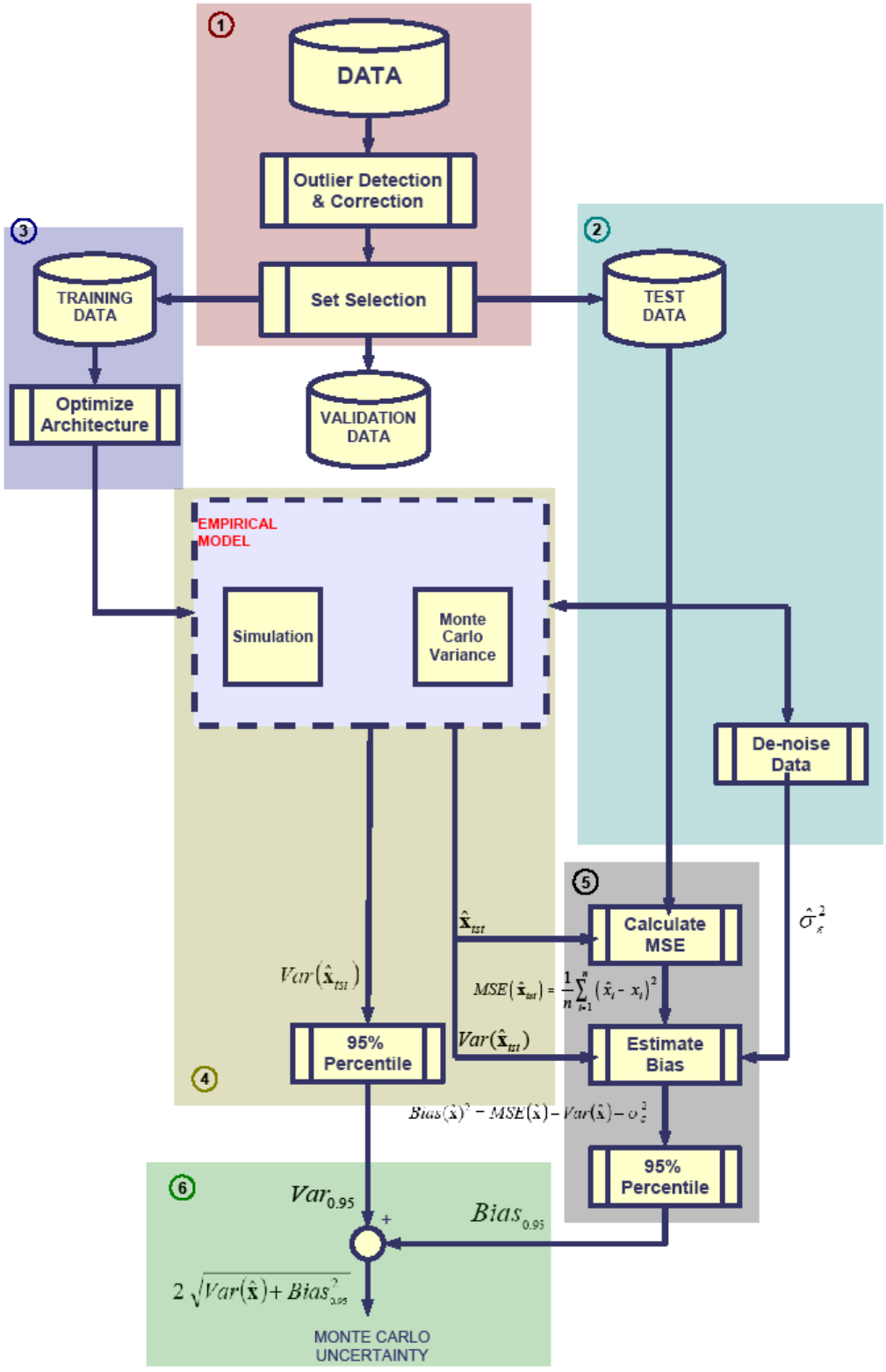


Fig. 6-2. Process diagram for Monte Carlo uncertainty estimation of an empirical model.

6.3.3 Fault detection

Because the overall goal of estimating an empirical model's uncertainty for instrument calibration verification is to identify sensor faults, a methodology must be developed that provides fault detection capability in an easily tractable manner. The method developed and employed in this example is termed error uncertainty limit monitoring (EULM). Rather than simply thresholding the sensor's error, the predicted uncertainty (i.e., residual CI) is monitored, and a fault is registered when the uncertainty interval crosses the specified threshold. This detection method is more clearly stated by the following procedure.

1. Estimate the residual CI by denoising the residuals and applying an uncertainty estimate.
2. Define a calibration threshold (assume it is 1%).
3. Identify a sensor as out of calibration if sustained uncertainty estimates for the denoised residuals cross the $\pm 1\%$ threshold.

6.4 Example with Artificially Generated Data

For this example, data were generated with the following base equation:

$$\boldsymbol{f}(\boldsymbol{x}) = \sin\left(2\pi\sqrt{\boldsymbol{t}}\right) . \tag{6.4.1}$$

Here, $\boldsymbol{t}$ is uniformly spaced on the interval of [0, 3]. To provide data that is similar to that of redundant sensors, three unique biases were added. The following equation provides the true data for the three artificial variables used in this section:

$$\boldsymbol{f}_{\boldsymbol{i}}(\boldsymbol{x}) = \sin\left(2\pi\sqrt{\boldsymbol{t}}\right) \quad \text{with } \boldsymbol{i} = 1,2,3 . \tag{6.4.2}$$

Finally, normally distributed random noise with a mean of 0 and a variance of 0.04, $\boldsymbol{N}_{\boldsymbol{i}}$ (0, 0.4), was added to each variable's true value.

$$\boldsymbol{x}_1 = \boldsymbol{f}_1(\boldsymbol{x}) + \boldsymbol{N}_1(0, 0.04) . \tag{6.4.3}$$

$$\boldsymbol{x}_2 = \boldsymbol{f}_2(\boldsymbol{x}) + \boldsymbol{N}_2(0, 0.04) . \tag{6.4.4}$$

$$\boldsymbol{x}_3 = \boldsymbol{f}_3(\boldsymbol{x}) + \boldsymbol{N}_3(0, 0.04) . \tag{6.4.5}$$

One such signal is presented in Fig. 6-3.

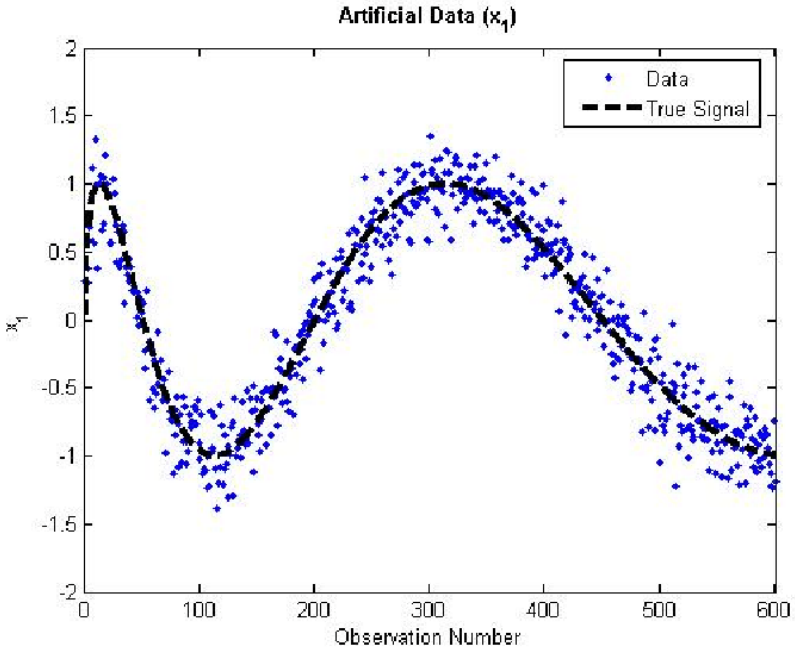


Fig. 6-3. Artificially generated signal.

The data sets were selected by first creating 10 sets of 60 continuous observations. Next, the training data were defined to be sets 1, 5, 6, and 10. The test data are defined to be sets 2, 3, and 4, while the validation data are defined to be sets 7, 8, and 9. The selected sets are presented in Fig. 6-4. The training data were used to develop the empirical models, while the test data were used to develop estimates for the noise variance and the model's bias. Finally, the validation data were used to validate the uncertainty estimates by calculating coverage values.

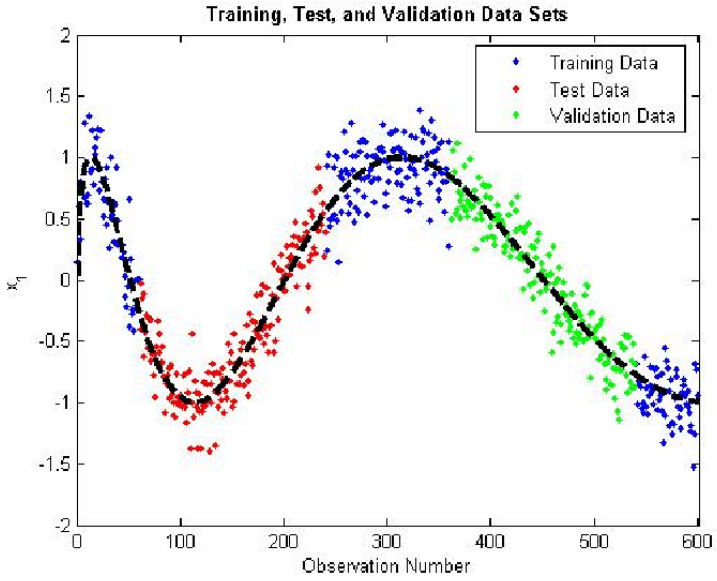


Fig. 6-4. Training, test, and validation data sets.

Before the empirical model's uncertainty is estimated, the assumption of normally distributed random error is validated by using wavelet denoising to estimate the data's true value. In Fig. 6–3, it can be seen that wavelet denoising is able to estimate the data's true value to a high degree of accuracy. Furthermore, it can be seen in Table 6-1 that the estimated noise parameters are very near their actual values.

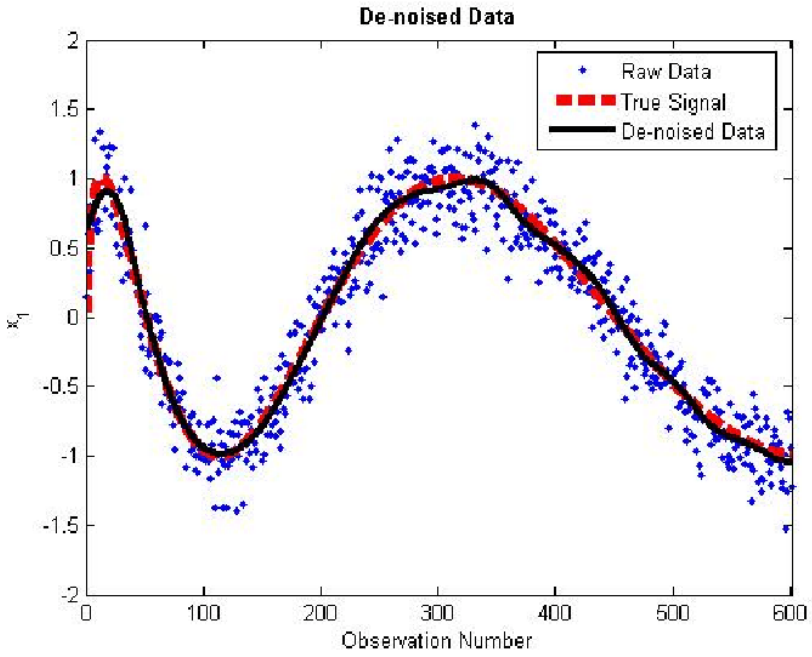


Fig. 6-5. Wavelet denoising estimates for the true values of the artificial data.

Table 6-1. Comparison of estimated noise parameters to actual distribution

		x_1	x_2	x_3
Mean (μ)	**Actual**	–0.0031	0.0003	–0.0011
	Estimated	–0.0005	–0.0005	–0.0004
STD (σ)	**Actual**	0.1925	0.1970	0.2060
	Estimated	0.1913	0.1975	0.2060

The estimated standard deviations in Table 6-1 were squared to provide $\hat{\sigma}_\varepsilon^2$ for use in Eq. (6.1).

6.4.1 Analytic methodologies

To test the analytic equations developed in Chap. 5, representative models were created and tested. For completeness, the AAKR model was constructed by using 60 of the training observations as memory vectors and using a bandwidth of 0.5. The AAMSET model was constructed by selecting 12 memory

vectors from the training data according to a min-max and sort-select strategy described by Gross et al. [2002] and uses a bandwidth of 1.0. Finally, the AANN model architecture was selected to have one mapping/demapping neuron and one bottleneck neuron.

The contributing factors to the total uncertainty are presented in Table 6-2, where the listed variances are the mean or expected variances.

Table 6-2. Contributing uncertainty factors for analytic uncertainty estimates of the artificial data

		x_1	x_2	x_3
	$\hat{\sigma}_\varepsilon^2$	0.0369	0.0346	0.0474
AAKR	**Variance**	0.0017	0.0021	0.0031
	Bias	0.0000	0.0048	0.0000
AAMSET	**Variance**	0.0044	0.0069	0.0078
	Bias	0.0199	0.0341	0.0093
AANN	**Variance**	0.0005	0.0006	0.0005
	Bias	0.0000	0.0000	0.0000

The mean uncertainty of each model for an approximate 95% confidence level was calculated and is displayed Table 6-3. Also, the mean coverages of the estimated prediction intervals were found to have a comparable performance to the theoretical 95%. This indicates that the analytic uncertainty estimates are near their true values.

Table 6-3. Analytic uncertainty estimates and Pl coverages for the artificially generated data

	$U(x_1)$	$U(x_2)$	$U(x_3)$	**Mean coverage**
AAKR	0.422	0.421	0.463	0.948
AAMSET	0.463	0.47	0.44	0.939
AANN	0.424	0.404	0.41	0.981

Finally, the model predictions and prediction uncertainties are displayed in Fig. 6-6. For each of the models, the raw data are displayed (blue points) with their respective estimates and 95% uncertainties (red error bars).

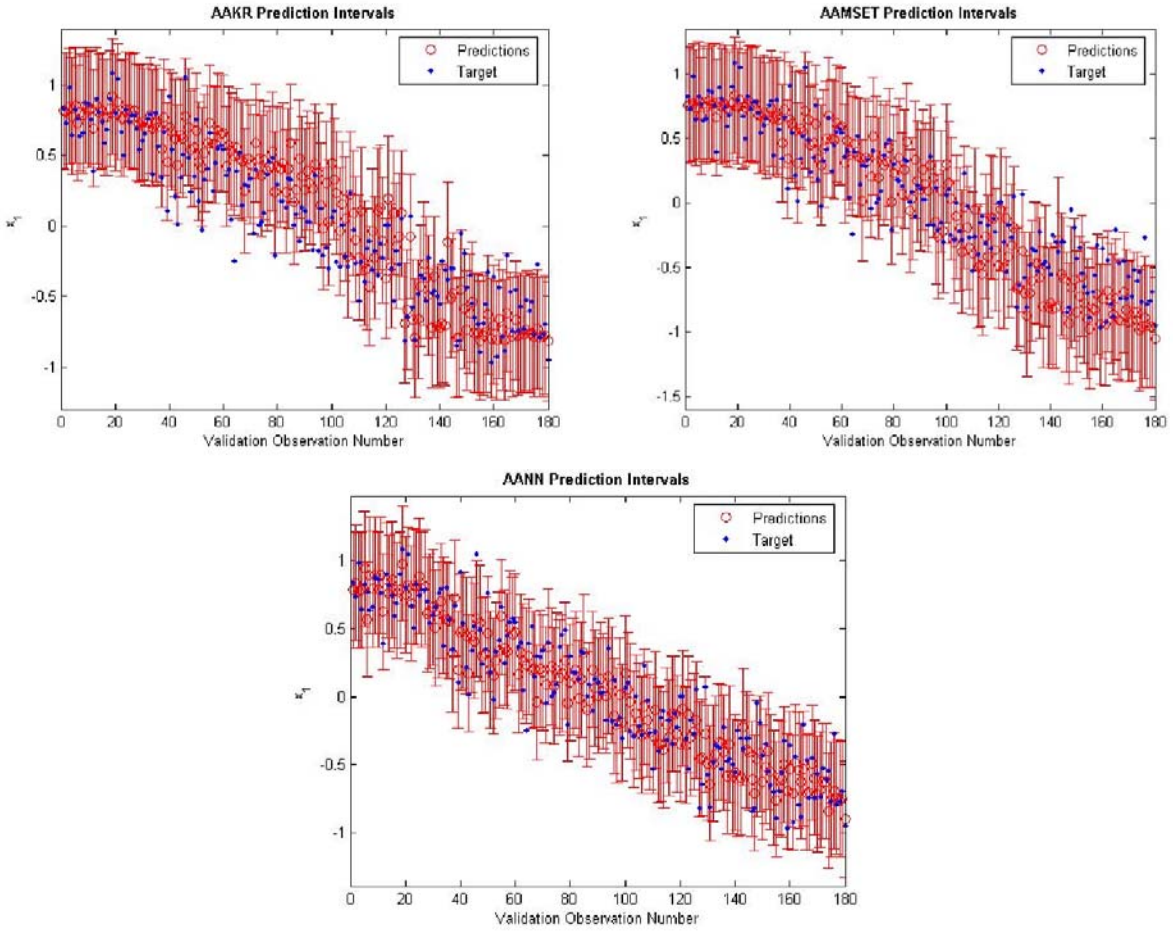


Fig. 6-6. Predications and analytic uncertainty estimates of the three empirical models for the artificially generated data.

Notice that the uncertainty is dominated by the noise variance (Fig. 6-7), whose percent contribution is on the order of 80–95% of the total uncertainty. If the noise variance is large, the model's ability to detect subtle sensor drifts will be difficult when using PIs on the predictions. As was discussed earlier, CIs on filtered residuals circumvent that problem. This issue will be addressed in the subsequent sections in which actual plant data are analyzed.

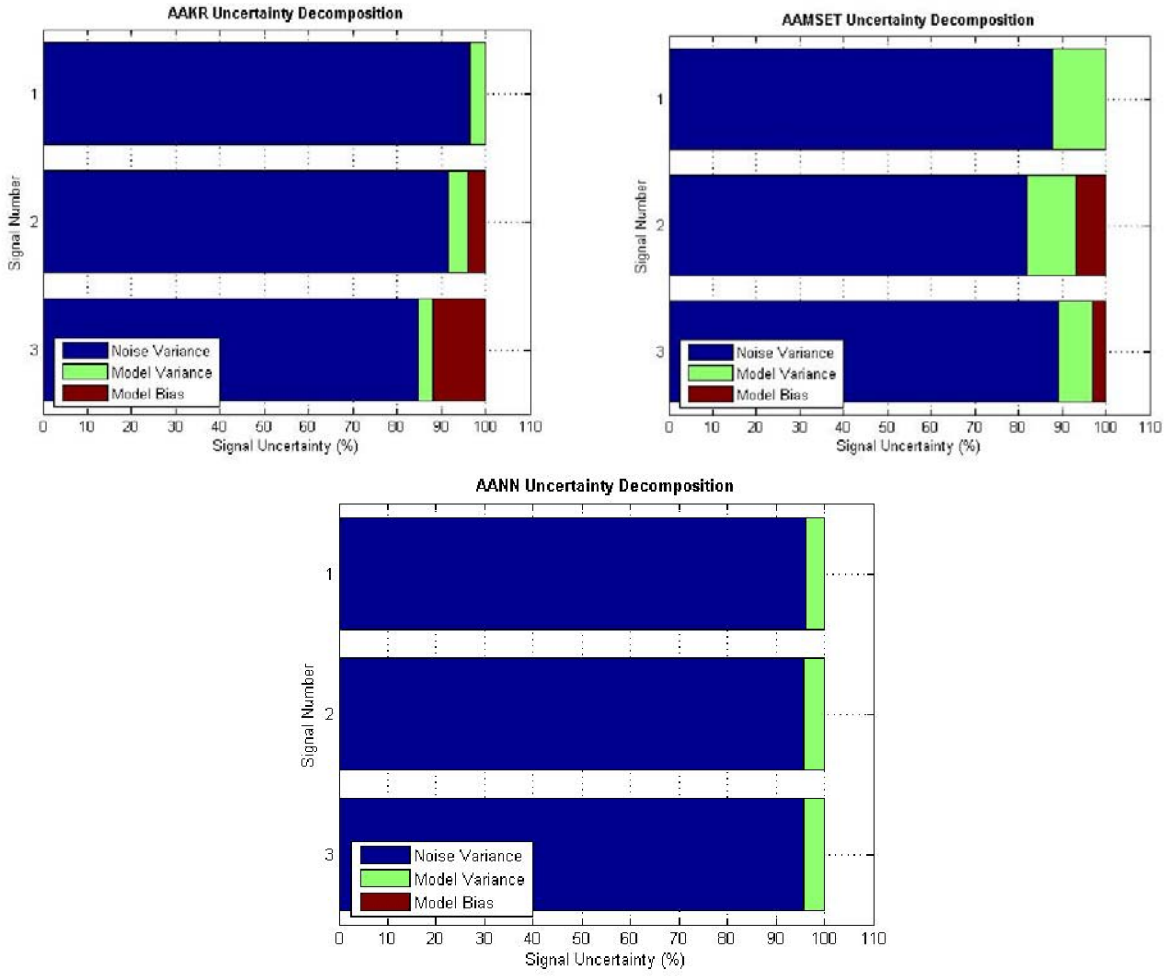


Fig. 6-7. Prediction interval uncertainty decomposition for analytic estimates of the artificially generated data.

Overall, the uncertainty estimates are comparable for each empirical modeling technique. It can be seen that because the AAMSET restricts the number of memory vectors used in the model, its uncertainty is larger than the AAKR model, which uses more exemplar vectors. However, if the number of memory vectors in the AAKR model is restricted this difference will not be significant. It is common for the AAMSET to have very few exemplar vectors to minimize the possible negative effects of inverting an ill-conditioned similarity matrix. If more vectors are used, the observations become similar, and ill-conditioning results. Regularization methods can be used to produce more stable results when many prototype vectors are used [Hines 2002].

6.4.2 Monte Carlo methodologies

To compare the Monte Carlo uncertainty estimation methodologies, the AAKR model described in the previous section was used with the two Monte Carlo algorithms described in Chap. 6. This section will compare the techniques according to their asymptotic performance and convergence characteristics.

6.4.2.1 Comparison of asymptotic performance

The Monte Carlo process for estimating the uncertainty of the model's expected predictions for the test data was generated as follows:

1. create prototype training data:
   (a) direct bootstrap—randomly sample the training data with replacement,
   (b) LHS—sample the estimated noise distribution and add sample to the estimated noise free training data;
2. construct prototype model,
3. simulate model with test data,
4. store results,
5. return to 1 until 250 prototype models have been created and tested, and
6. estimate the model's prediction variance.

Because Monte Carlo methodologies are used to estimate a single-point uncertainty estimate for the data, the variance is estimated by the 95% percentile of the estimated distribution and then expanded such that it has the same dimension as the validation data. Next, the single-point uncertainty for the validation data is estimated by combining the mean prediction variance with the estimated noise variance and the bias. Finally, the AAKR model is used to make predictions for the validation, which are used to construct the PIs.

The contributing factors to the model uncertainty are presented in Table 6-4. The estimated single-point uncertainties with the expected PI coverages are presented in Table 6-5. Finally, the error bar uncertainty plots of the results are presented in Fig. 6-8. Note that the bias values quoted in Table 6-4 are not the variable's bias, but the respective model prediction bias.

Table 6-4. Contributing uncertainty factors for Monte Carlo uncertainty estimates of the artificial data and AAKR model

		x_1	x_2	x_3
	$\hat{\sigma}_\varepsilon^2$	0.0369	0.0346	0.0474
Actual (analytic)	***Var***$(\hat{\mathbf{x}})$	0.0017	0.0021	0.0031
	Bias	0.0000	0.0048	0.0000
Bootstrap	***Var***$(\hat{\mathbf{x}})$	0.0046	0.0060	0.0048
	Bias	0.0000	0.0000	0.0000
LHS	***Var***$(\hat{\mathbf{x}})$	0.0038	0.0042	0.0043
	Bias	0.0000	0.0014	0.0000

Table 6-5. Monte Carlo uncertainty estimates and PI coverages for artificial data

	$U(x_1)$	$U(x_2)$	$U(x_3)$	Mean coverage
Actual (analytic)	0.422	0.421	0.463	0.948
Bootstrap	0.453	0.441	0.463	0.961
LHS	0.449	0.427	0.463	0.957

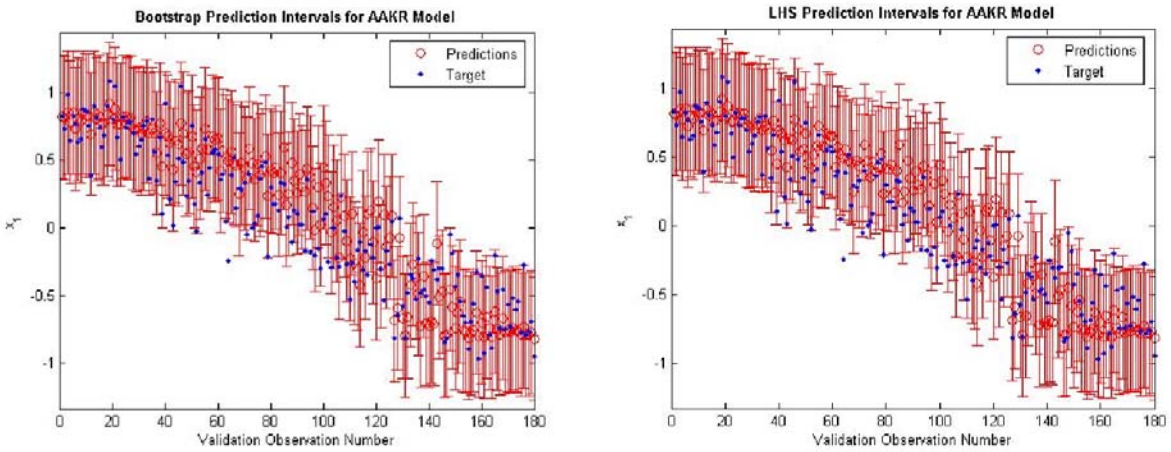


Fig. 6-8. Predictions and Monte Carlo uncertainties for artificial data.

It can be seen in Table 6-5 and Fig. 6-6 that the methods are almost equivalent. Also, it can be seen that the estimated uncertainties and their respective PI coverages are very similar to the analytic estimates. This indicates that for a sufficiently large number of Monte Carlo samples, the methods produce similar results to the analytic equations, thus validating the theoretical derivation of the equations presented in Sect. 4.2. Also, notice in Fig. 6-9 that the major contributing factor to uncertainty is the noise variance. In the next section, the convergence characteristics of the two Monte Carlo methods will be evaluated.

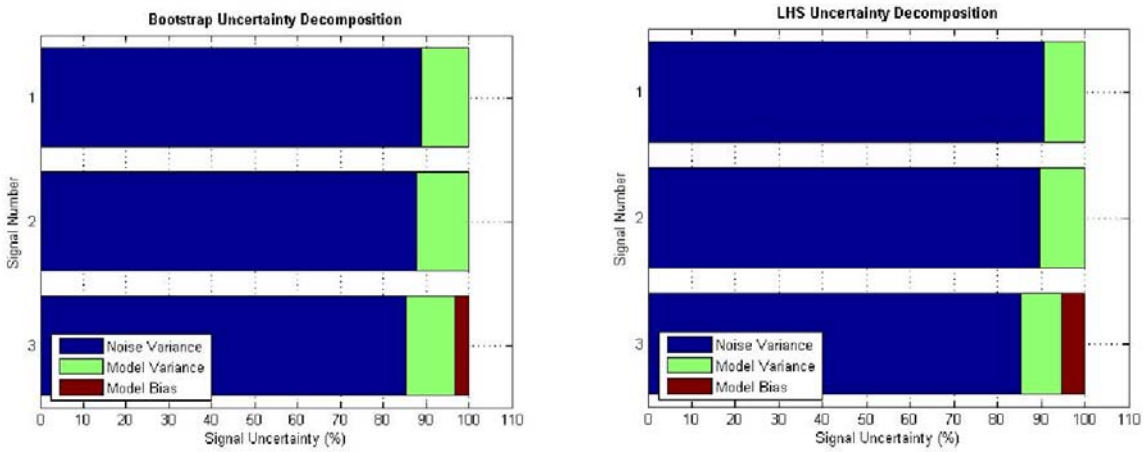


Fig. 6-9. Prediction interval uncertainty decomposition for Monte Carlo estimates of the artificially generated data.

6.4.2.2 Comparison of convergence characteristics

In Sect. 5.3, the convergence properties of the two Monte Carlo estimation methods were compared with simulated data. In this section, actual nuclear power plant data will be used. The process used to characterize the convergence characteristics of the two Monte Carlo estimation methods is as follows:

1. Estimate uncertainty of the AAKR model by creating and testing 2, 5, 10, 20, 30, 40, 50, 75, and 100 prototype models;
2. Store the estimated variance for the 10th observation of the 1st variable;
3. Return to step 1 until 100 estimates have been made.

The result of the above algorithm is a distribution of the estimated variance as a function of the number of prototype models. It is expected that for each Monte Carlo sampling method, the uncertainty of the estimated variances would start high, since only two prototype models are tested, and will begin to decrease as more prototype models are tested. The main purpose of this methodology is to ascertain whether or not the more complex LHS algorithm converges faster than the direct bootstrap for actual data. In other words, this comparison answers the question whether or not the increased number of assumptions required for LHS produce any positive effects on the number of model's required to reach the stability of the simple, direct bootstrap.

Before results are presented, the technique used to ascertain the variance distribution's uncertainty must be discussed. For this analysis, the uncertainty of the variance is defined as the interval of values between the 2.5 and 97.5 percentiles. For example, this interval is defined as approximately $\pm 2\sigma$ for the standard normal distribution (i.e., $\boldsymbol{N}(0,1)$), which is presented in Fig. 6-10.

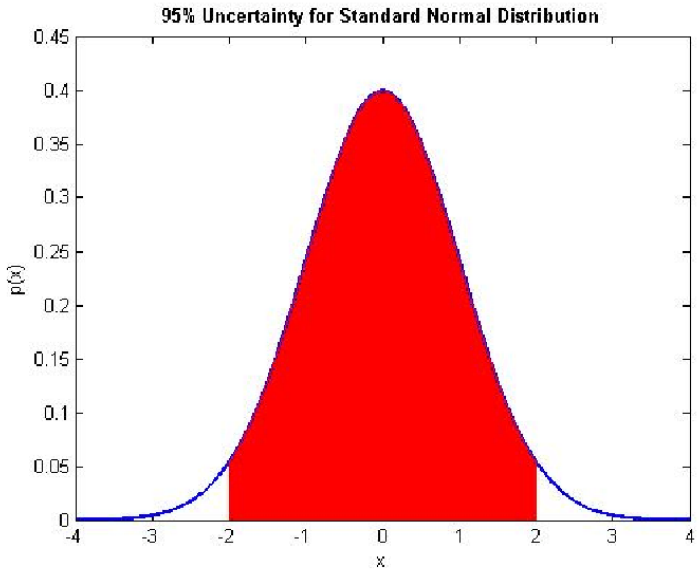


Fig. 6-10. 95% uncertainty for standard normal distribution.

Because the distribution of the estimated variances is not known, the percentiles are calculated, via a numerical process, as follows:

1. order the 100 variance estimates;
2. define the 2.5% percentile's $\boldsymbol{x}$ value ($\boldsymbol{x}_{.025}$) as the average of the 2nd and 3rd variance estimate from the ordered values;
3. define the 97.5% percentile's $\boldsymbol{x}$ value ($\boldsymbol{x}_{.975}$) as the average of the 97th and 98th variance estimate from the ordered values; and
4. define the 95% uncertainty as the difference between $\boldsymbol{x}_{.975}$ and $\boldsymbol{x}_{.025}$ ($\boldsymbol{x}_{.975}$–$\boldsymbol{x}_{.025}$).

The median variance estimates, their respective percentiles, and the 95% uncertainties for the direct bootstrap and LHS algorithms may be seen in Table 6-6.

Table 6–6. Convergence characteristics for direct bootstrap and LHS sampling methods and artificial data

	Number of prototype models	2.5 Percentile	Median variance	97.5 Percentile	95% Uncertainty
Bootstrap	2	0	0.0024	0.0272	0.0272
	5	0.0009	0.0057	0.0199	0.019
	10	0.0016	0.0063	0.0142	0.0126
	20	0.003	0.0068	0.0143	0.0113
	30	0.0043	0.0066	0.0118	0.0075
	40	0.0039	0.0065	0.0112	0.0073
	50	0.0041	0.0071	0.013	0.0089
	75	0.0052	0.0074	0.0104	0.0052
	100	0.005	0.0072	0.0092	0.0042
LHS	2	0.0001	0.0019	0.0276	0.0276
	5	0.0004	0.0058	0.0192	0.0188
	10	0.0018	0.0055	0.0142	0.0124
	20	0.0021	0.0063	0.0117	0.0096
	30	0.0033	0.0055	0.0098	0.0066
	40	0.0037	0.0061	0.0085	0.0048
	50	0.0038	0.0061	0.0113	0.0075
	75	0.004	0.0059	0.0098	0.0059
	100	0.0045	0.0062	0.0081	0.0036

The behavior of the variance uncertainty is presented in Fig. 6-11.

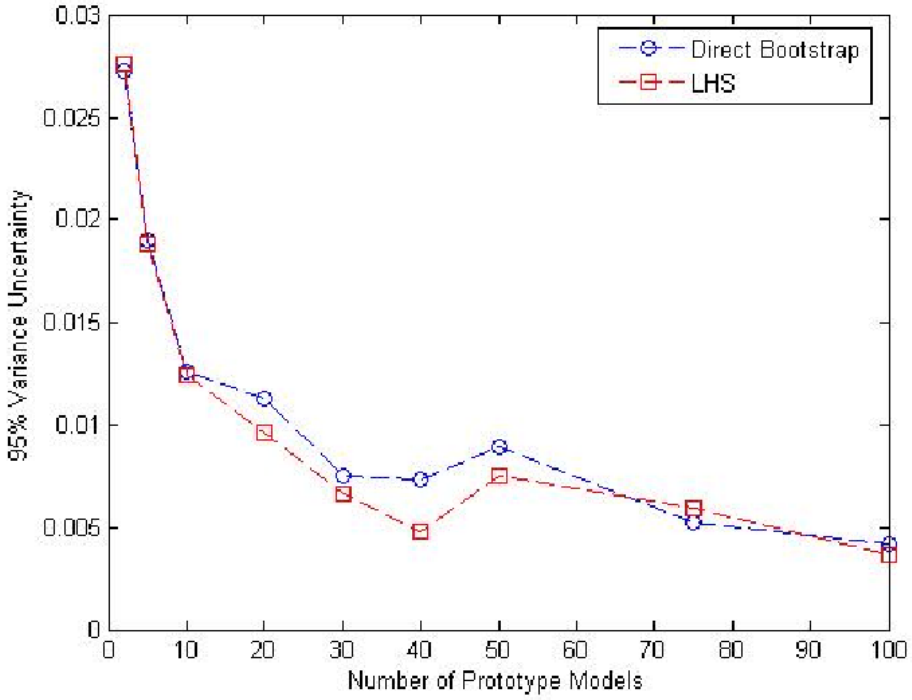


Fig. 6-11. Convergence characterization of Monte Carlo variance estimates for artificial data.

It can be seen that the LHS and direct bootstrap convergence rates are not significantly different. Therefore, it can be inferred that the convergence performance of LHS is not sufficiently improved to warrant the increase in computational complexity as compared to the direct bootstrap. These results are slightly different than those presented in Sect. 5.3, but a more exact method is used to quantify the estimated variance.

6.5 Example with Nuclear Plant Data

In this section, data collected from the steam system of an operating nuclear plant is used to test the analytic and Monte Carlo uncertainty estimation methodologies. Next, error uncertainty limit monitoring (EULM) is used to detect an artificial sensor drift of 1%.

The model (variable grouping) chosen was developed during the EPRI OLM Implementation Project and is currently being used to monitor steam system sensor calibration at an operating plant; thus, it is an appropriate model to evaluate. The steam system model contains 13 plant sensors, primarily from 1 loop, which include 2 feedwater flow sensors, 2 steam flow sensors, 3 steam generator level sensors, 2 turbine pressure sensors, and 3 steam pressure sensors. The quoted sensor units are as follows: (1) feedwater and steam flow in thousands of pounds per hour (klbm/h) was logged by the data acquisition system as KBH, (2) steam generator level in percent (i.e., 100% is full), (3) turbine pressure in pounds per square inch atmospheric (PSIA), and (4) steam pressure in pounds per square inch gauge (PSIG). The training data for each of the sensor types is presented in Fig. 6-12.

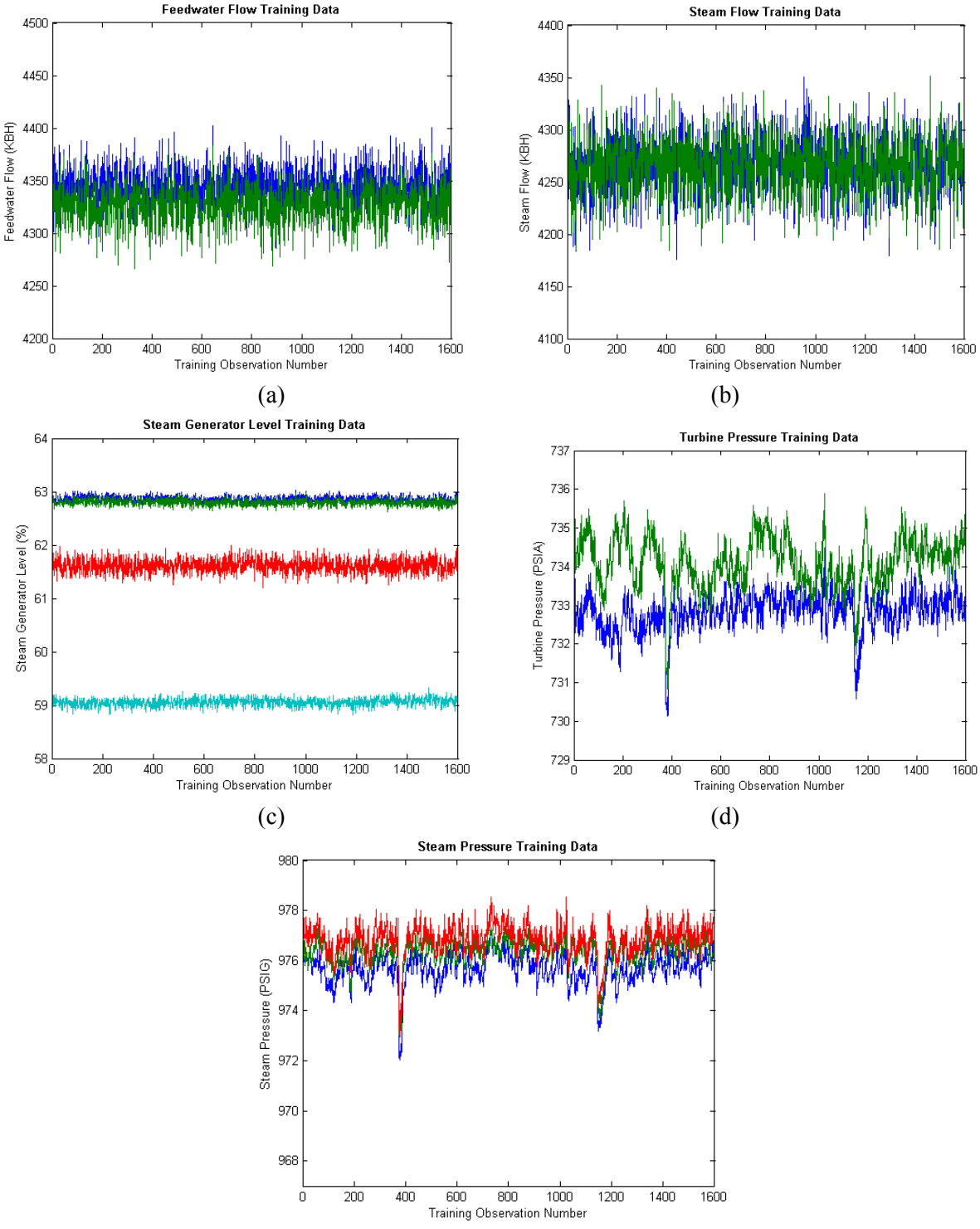


Fig. 6-12. Training data for (a) feedwater flow, (b) stream flow, (c) stream generator level, (d) turbine pressure, and (e) steam pressure.

The data presented in Fig. 6-12 were selected from data collected every 2 min over a 2-month period. Overall the training, test, and validation data span approximately 2 weeks of data observing every 5th sample or every 10 min.

The training data were chosen to be 1600 observations from steady state plant operation. The test and validation data were chosen to be two successive sets of 200 observations sampled from steady state plant operation. The training data were used to develop the empirical models, the test data were used to develop estimates for the noise variance and the model's bias, and the validation data were used to validate the uncertainty estimates and test the empirical model's ability to detect an artificial drift.

To begin this analysis, the test data were denoised with the wavelet technique described earlier. The resulting signals are presented in Fig. 6-13. The estimated noise variances are presented in Table 6-7. Table 6-7 also includes 2 standard deviations of the noise estimates in terms of percentage of the sensor's mean value. For example, if a sensor's nominal value is 100 and 2 standard deviations of the noise distribution is estimated to be 10, then the standard deviations in terms of the nominal value is $100\% \times \frac{10}{100} = 10\%$. In other words, 95% of the noise is on the interval of $\pm 10\%$ the signal's nominal value.

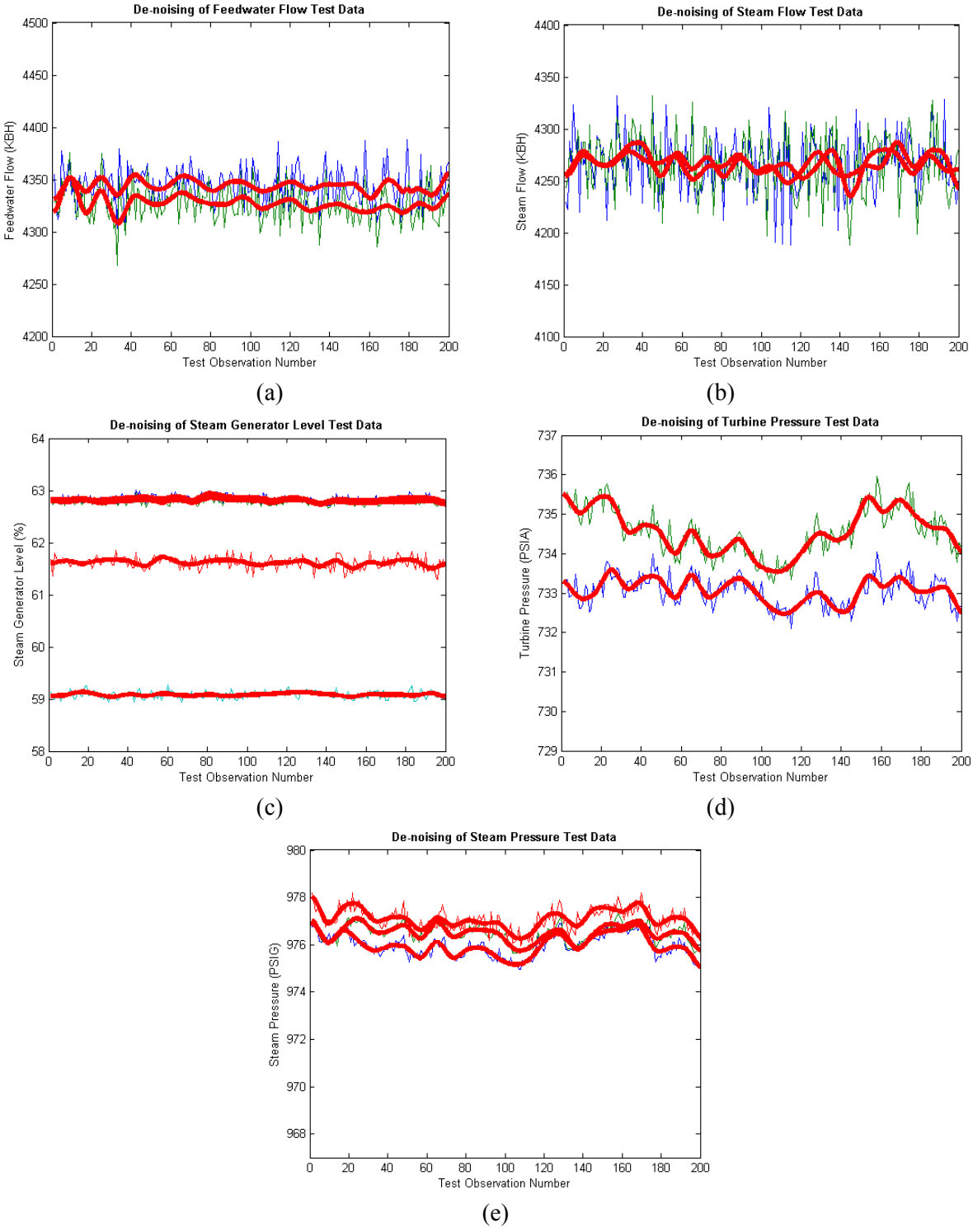


Fig. 6-13. Denoised test data for (a) feedwater flow, (b) steam flow,(c) steam generator level, (d) turbine pressure, and (e) steam pressure.

Table 6-7. Estimated noise variance and noise level for nuclear plant data

	Feedwater Flow		Steam Flow		Steam Generator Level				Turbine Pressure		Steam Pressure		
	#1	#2	#1	#2	#1	#2	#3	#4	#1	#2	#1	#2	#3
Noise Variance	298.47	298.20	779.89	792.27	0.05	0.02	0.03	0.01	0.13	0.14	0.18	0.20	0.15
2 STDs (%)	0.80	0.80	1.31	1.32	0.73	0.50	0.52	0.36	0.10	0.10	0.09	0.09	0.08

In Fig. 6-13 notice that the wavelet denoising algorithm may have over smoothed the data. This supposition is based upon the features of the plots presented in Fig. 6-13. It can be seen that on numerous occasions the data for each of the variables within the groups move well outside the range of values that the wavelet denoising algorithm deems important. The reason that this occurs is that the wavelet denoising algorithm examines all of the sensors in the data set as a group and does not produce optimal filtering parameters for each sensor. This effectively degrades the algorithm's ability to identify local group correlations because a significant correlation only occurs when the behavior is present in the observations of most of the sensor's data. For example, even though the correlated spikes in the turbine pressure data occur in both sensors, the wavelet denoising algorithm is unable to identify these observations as containing useful system information because the same pattern is not present in the other sensor groups. From these results, the coverages of the estimated CIs are expected to deviate from their theoretical value of 0.95, because the estimated noise variance is expected to be biased. Future work in this area should allow for wavelet denoising to be applied to each sensor in the group, which would improve the accuracy of the estimated "true" signals and the noise distribution. This is discussed in more detail in Volume 3 of this NUREG series.

Next, notice that the largest noise level is on the feedwater flow and steam flow sensors. For two standard deviations, the sensor uncertainties are on the order of ±1%, which are the specified calibration limits. Because the variability in the feedwater flow is one of the largest, these sensors will be one of the more difficult groups to successfully apply the calibration monitoring methods described in this report. Therefore, most of the discussion in the remainder of this example will focus on the two feedwater flow sensors.

6.5.1 Analytic methodologies

The AAKR model used in this example is composed of 200 memory vectors, has a kernel bandwidth of 0.5, and uses the L1-norm distance function. The AAMSET model is composed of 52 memory vectors, has a kernel bandwidth of 1.0, and uses the L1-norm distance function. Finally, the AANN model contains six mapping/demapping neurons and three bottleneck neurons.

As in the previous example, the analytic CIs were used to estimate the model uncertainty. The contributing factors to uncertainty are presented in Table 6-8. The mean uncertainties and their respective CI coverages of the denoised data are presented in Table 6-9 and Table 6-10, respectively. The point-wise uncertainty estimates for the first feedwater flow sensor are displayed in Fig. 6-14, where the raw data are denoted by blue points, the denoised data by a dashed black line, and the prediction CIs by red error bars. Finally, the denoised residuals and their uncertainties are presented in Fig. 6-15, where the raw residuals are denoted by blue points, the CIs of the denoised residuals by red error bars, and the ±1% tolerances by heavy dashed black lines.

Table 6-8. Contributing analytic uncertainty estimate components for the nuclear plant data

		Feedwater Flow		Steam Flow		Steam Generator Level				Turbine Pressure		Steam Pressure		
		#1	#2	#1	#2	#1	#2	#3	#4	#1	#2	#1	#2	#3
AAKR	Variance	127.34	119.64	396.15	336.74	0.00	0.00	0.01	0.00	0.08	0.12	0.07	0.06	0.08
	Bias	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AAMSET	Variance	168.27	195.78	410.13	554.01	0.00	0.00	0.01	0.00	0.08	0.17	0.08	0.05	0.09
	Bias	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AANN	Variance	5.30	5.82	25.64	17.91	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00
	Bias	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 6-9. Analytic uncertainty estimates for the nuclear plant data

		Feedwater Flow		Steam Flow		Steam Generator Level				Turbine Pressure		Steam Pressure		
		#1	#2	#1	#2	#1	#2	#3	#4	#1	#2	#1	#2	#3
AAKR	Uncert	22.52	21.83	39.72	36.62	0.06	0.06	0.19	0.11	0.58	0.69	0.54	0.48	0.56
	Uncert (%)	0.52	0.50	0.93	0.86	0.09	0.10	0.31	0.18	0.08	0.09	0.06	0.05	0.06
AAMSET	Uncert	25.79	27.82	40.26	46.79	0.09	0.08	0.22	0.12	0.55	0.79	0.56	0.45	0.60
	Uncert (%)	0.59	0.64	0.94	1.10	0.14	0.13	0.35	0.20	0.07	0.11	0.06	0.05	0.06
AANN	Uncert	4.61	4.83	10.13	8.46	0.01	0.01	0.01	0.00	0.16	0.24	0.13	0.13	0.14
	Uncert (%)	0.11	0.11	0.24	0.20	0.01	0.01	0.01	0.01	0.02	0.03	0.01	0.01	0.01

Table 6-10. Coverages of denoised data by analytic CI

	Feedwater Flow		Steam Flow		Steam Generator Level				Turbine Pressure		Steam Pressure			Mean Coverage
	#1	#2	#1	#2	#1	#2	#3	#4	#1	#2	#1	#2	#3	
AAKR Coverage	0.89	0.89	0.96	0.90	0.72	0.76	0.92	0.90	0.95	0.89	0.89	0.92	0.93	0.88
AAMSET Coverage	0.90	0.88	0.94	0.96	0.93	0.92	0.89	0.91	0.94	0.90	0.88	0.91	0.91	0.91
AANN Coverage	0.19	0.24	0.35	0.32	0.06	0.08	0.04	0.02	0.47	0.52	0.35	0.46	0.33	0.26

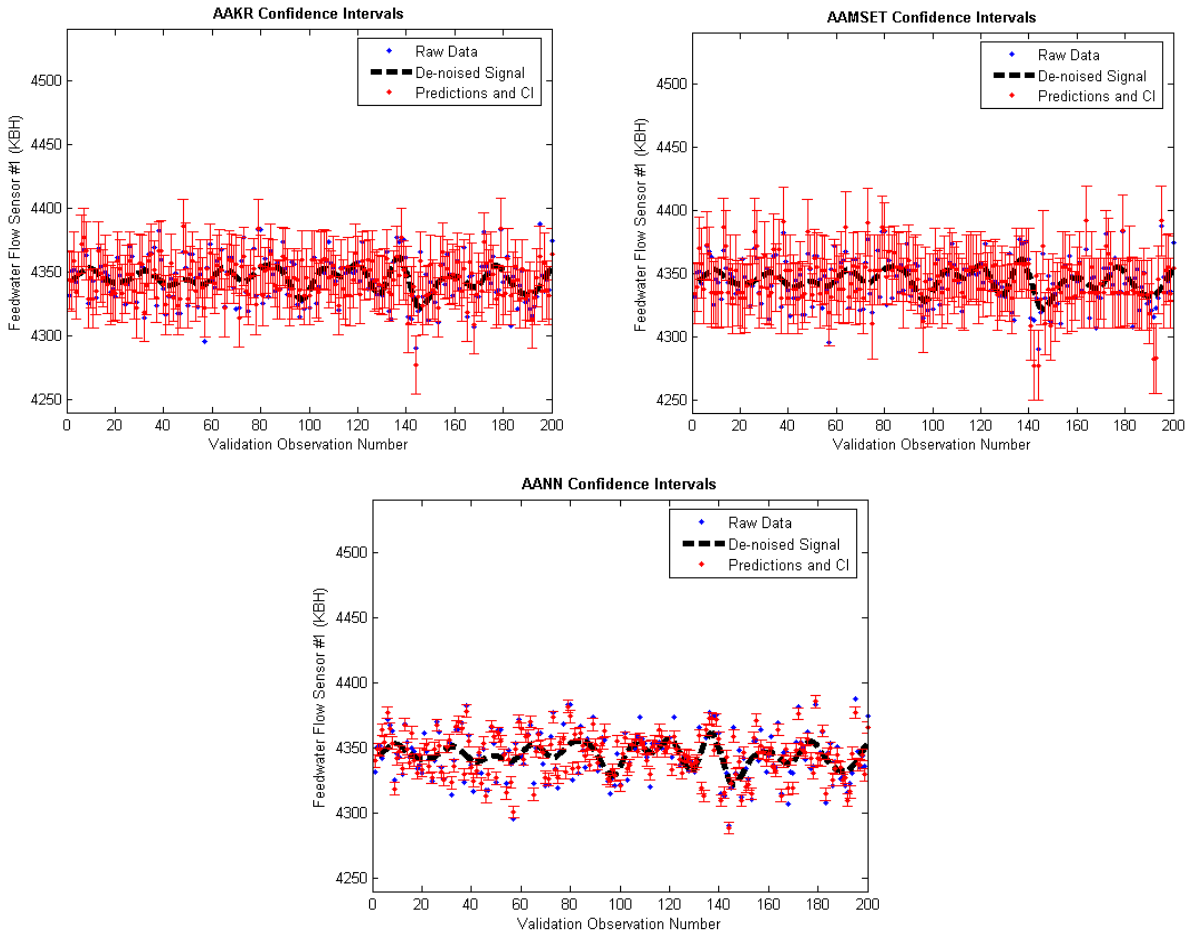


Fig. 6-14. Analytic CIs for AAKR, AAMSET, and AANN models for the first feedwater flow sensor.

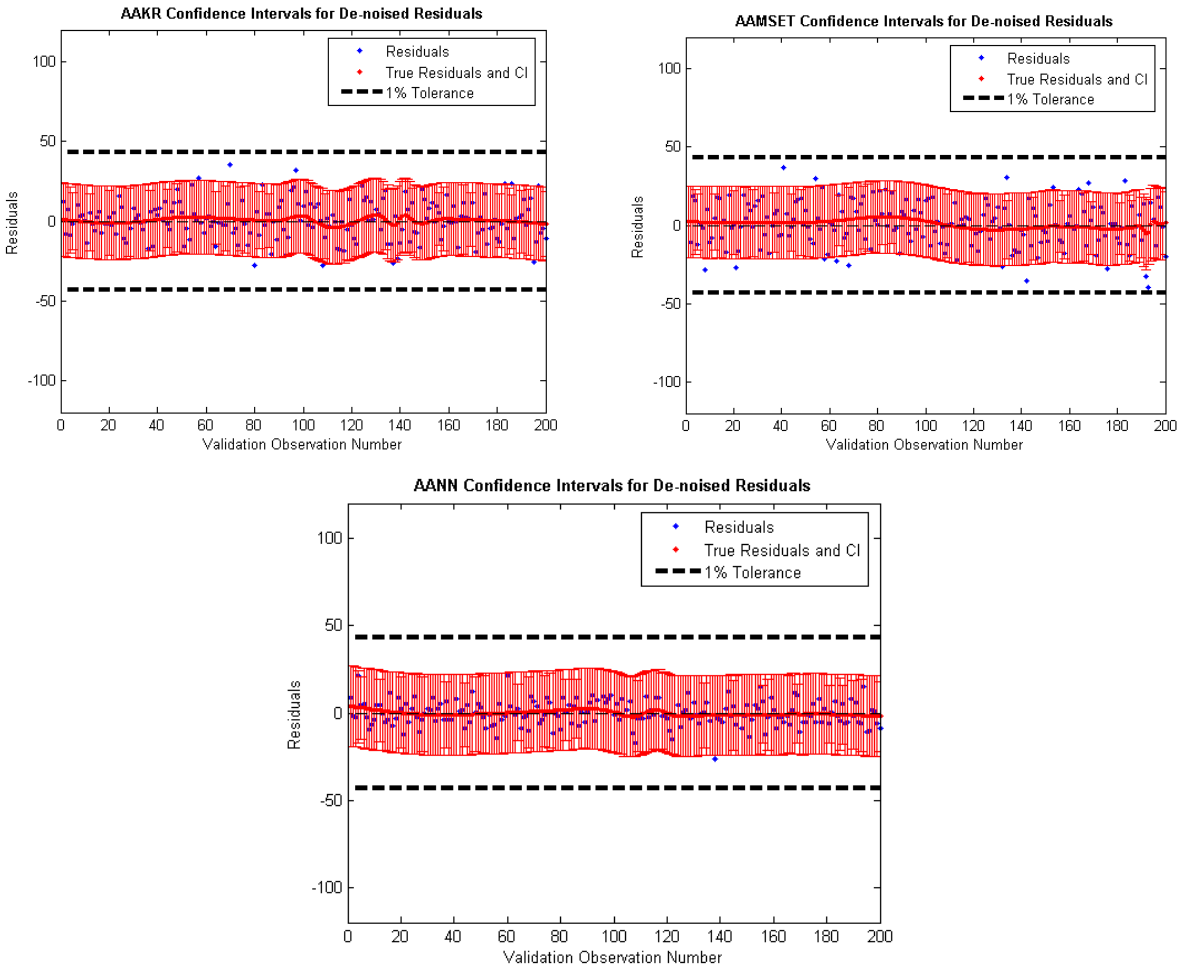


Fig. 6-15. Denoised residuals and analytic uncertainty estimates for the first feedwater flow center.

It can be seen in Table 6-8 that the major contributing factor to model uncertainty is the model variance. This is a result of the fact that the data are relatively simple, in that they generally oscillate about a nominal value and contain a single state for the operating reactor (i.e., ~100% power). Data that are more varied may have larger bias values.

Next, it can be seen in Table 6-9 that the uncertainties are small compared to the estimated noise levels in Table 6-7. For example, the largest sensor uncertainty for the AAKR model is 0.93% its nominal value, while the noise level is 1.31%. Furthermore, notice that the "worst case" estimate of 0.93% is within the 1% tolerance. This proves that the presented analytic methodologies are applicable for use in instrument calibration validation systems; however, even the slightest drift would require recalibration.

It can be seen in Table 6-9 and Fig. 6-14 that the fraction of the denoised data covered by the analytic CIs is slightly below the theoretical 0.95 or 95% for the AAKR and AAMSET. Next, notice that the coverage of the AANN CIs is well below the theoretical 0.95. These features may be attributed to imperfections in the estimated "true" values. Another factor that may influence the CI coverage is the inherent difference between parameter estimates of the empirical model and wavelet denoising algorithm. In short, the empirical models may estimate a different "true" value than the wavelet denoising. One could argue for the validity of both methods' estimates. For example, the empirical model's estimates

could be considered more accurate since they base their estimates on historical, fault free data, while the wavelet denoising method could be considered more accurate since it acts to remove noise from data without making any suppositions as to the system's current state. At any rate, because the uncertainty of the model estimates are smaller than the noise level, the difference between the model's "perceived" true value and the denoised estimate affects the coverage of the analytic CIs.

Finally, notice that the denoised residuals and their analytic uncertainties are evenly distributed about zero for the AAKR and AAMSET model. Overall, the denoised residuals appear to behave as expected and therefore provide an accurate estimate of the model's noise free, prediction error. Also, the analytic residual uncertainties are mostly within the prescribed tolerance and therefore should provide accurate detectability for drifts on the order of ±1%.

6.5.2 Monte Carlo methodologies

To verify the analytic uncertainty estimates of the previous section, 250 prototype AAKR models were created and tested using both direct bootstrap sampling and LHS. The architecture of the AAKR models developed and tested in this section contains 200 memory vectors, has a bandwidth of 0.5, and uses the L1-norm distance function.

The contributing factors to uncertainty are presented in Table 6-11. The mean uncertainties and their respective CI coverages of the denoised data are presented in Table 6-11 and Table 6-12, respectively. The 95% percentile uncertainty estimates for the first feedwater flow sensor are displayed in Fig. 6-16, where the raw data are denoted by blue points, the denoised data by a dashed black line, and the prediction CIs by red error bars. Finally, the denoised residuals and their uncertainties are presented in Fig. 6-17, where the raw residuals are denoted by blue points, the CIs of the denoised residuals by red error bars, and the ±1% tolerances by heavy dashed black lines.

Table 6-11. 95% percentile of contributing uncertainty factors to Monte Carlo uncertainty estimates for nuclear plant data

		Feedwater Flow		Steam Flow		Steam Generator Level				Turbine Pressure		Steam Pressure		
		#1	#2	#1	#2	#1	#2	#3	#4	#1	#2	#1	#2	#3
Analytic	Variance	127.34	119.64	396.15	336.74	0.00	0.00	0.01	0.00	0.08	0.12	0.07	0.06	0.08
	Bias	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bootrstap	Variance	194.14	163.14	418.32	391.49	0.00	0.00	0.01	0.00	0.09	0.17	0.10	0.06	0.11
	Bias	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LHS	Variance	134.89	146.98	392.41	353.13	0.00	0.00	0.01	0.00	0.09	0.17	0.12	0.07	0.12
	Bias	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 6-12. 95% percentile Monte Carlo uncertainty estimates for nuclear plant data

		Feedwater Flow		Steam Flow		Steam Generator Level				Turbine Pressure		Steam Pressure		
		#1	#2	#1	#2	#1	#2	#3	#4	#1	#2	#1	#2	#3
Analytic	Uncert	22.52	21.83	39.72	36.62	0.06	0.06	0.19	0.11	0.58	0.69	0.54	0.48	0.56
	Uncert (%)	0.52	0.50	0.93	0.86	0.09	0.10	0.31	0.18	0.08	0.09	0.06	0.05	0.06
Bootstrap	Uncert	27.87	25.55	40.91	39.57	0.07	0.07	0.20	0.12	0.60	0.82	0.63	0.51	0.65
	Uncert (%)	0.64	0.59	0.96	0.93	0.11	0.11	0.33	0.20	0.08	0.11	0.07	0.05	0.07
LHS	Uncert	23.23	24.25	39.62	37.58	0.10	0.09	0.20	0.11	0.61	0.82	0.68	0.55	0.69
	Uncert (%)	0.53	0.56	0.93	0.88	0.16	0.15	0.32	0.19	0.08	0.11	0.07	0.06	0.07

Table 6-13. Coverages of denoised data by 95% percentile Monte Carlo CI

	Feedwater Flow		Steam Flow		Steam Generator Level				Turbine Pressure		Steam Pressure			Mean Coverage
	#1	#2	#1	#2	#1	#2	#3	#4	#1	#2	#1	#2	#3	
Analytic Coverage	0.89	0.89	0.96	0.90	0.72	0.76	0.92	0.90	0.95	0.89	0.89	0.92	0.93	0.88
Bootstrap Coverage	0.94	0.91	0.97	0.91	0.86	0.84	0.95	0.95	0.96	0.96	0.92	0.94	0.97	0.93
LHS Coverage	0.90	0.91	0.95	0.91	0.95	0.97	0.93	0.93	0.96	0.96	0.95	0.97	0.97	0.94

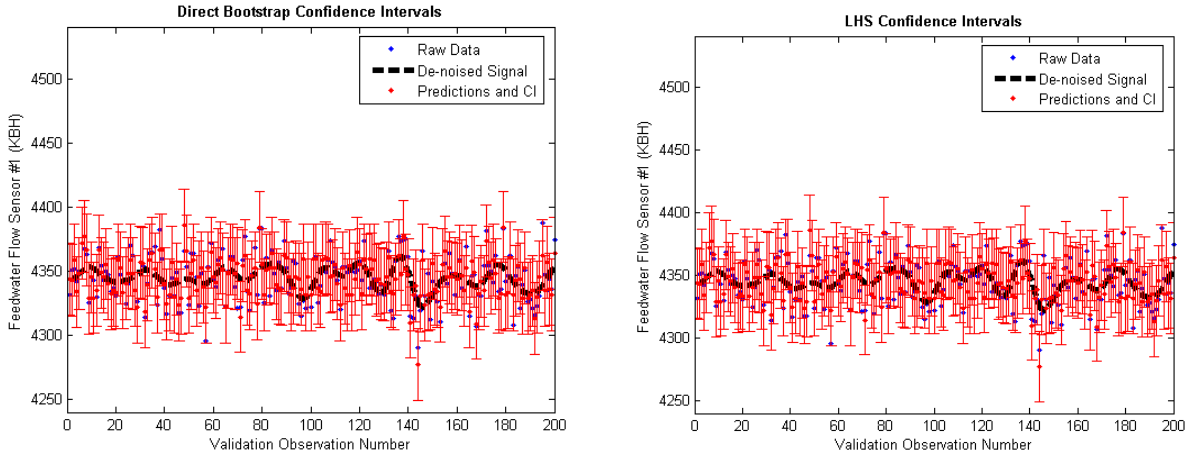


Fig. 6-16. 95% percentile Monte Carlo CIs for direct bootstrap sampling and LHS of the first feedwater flow sensor.

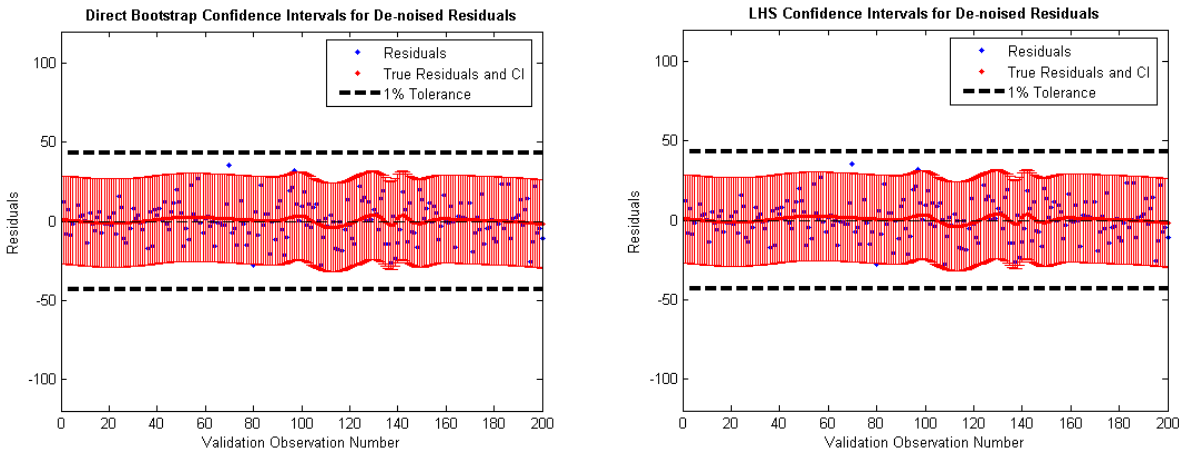


Fig. 6-17. Denoised residuals and Monte Carlo uncertainty estimates for the first feedwater flow sensor.

As with the analytic estimates, it can be seen in Table 6-11 that the major contributing factor to the Monte Carlo uncertainty estimate is the model variance. Furthermore, notice that Monte Carlo variance estimates are larger than the analytic estimates. This illustrates the fundamental difference between the two uncertainty estimation methodologies: the analytic methods estimate the uncertainty of a particular empirical model, while the Monte Carlo methods estimate the uncertainty of a particular model architecture for a given set of training and test data.

Next, notice in Table 6-12 that even though the Monte Carlo variance estimates are larger than the analytic estimates, the uncertainties are still within the ±1% tolerance. For example, the largest Monte Carlo uncertainty for the first feedwater flow sensor (direct bootstrap sampling) is ±27.87 klbm/h, which translates to ±0.94% of the sensor's nominal value.

It can also be seen in Table 6-12 and Fig. 6-16 that the coverage of the denoised data by the Monte Carlo CIs has increased as compared to the coverage of the analytic results and is now very near the theoretical value of 0.95. This increase in coverage is a direct result of the larger Monte Carlo uncertainty estimates.

Finally, Fig. 6-17 shows that the denoised residuals are evenly distributed about zero, and the uncertainties consistently lie within the 1% tolerance. Therefore, the Monte Carlo techniques should provide accurate fault detectability, despite an increase in predicted uncertainty as compared to the analytic estimates.

6.5.3 Drift detection

Recall that the overall goal of instrument calibration validation systems is to use the predictions and uncertainty to detect sensor drifts. As in most processes, sensors are allowed to operate about their nominal value with some tolerance. Recall that an error or residual tolerance of 1% was specified. Rather than simply thresholding the sensor's error, the predicted uncertainty was monitored, registering a fault when the uncertainty of the error moved outside the specified threshold. This method of fault detection is termed error uncertainty limit monitoring (EULM).

For this example, an artificial drift is introduced into the first feedwater flow sensor. This drift is set to progress linearly from 0% to −1% throughout the entire validation data. The normal, artificially drifted, and the AAKR predictions and uncertainties given the drifted data are presented in Fig. 6-18. The single-point uncertainty produced by direct bootstrap sampling and 250 prototype models was used, because its uncertainty estimates are the largest and therefore most conservative.

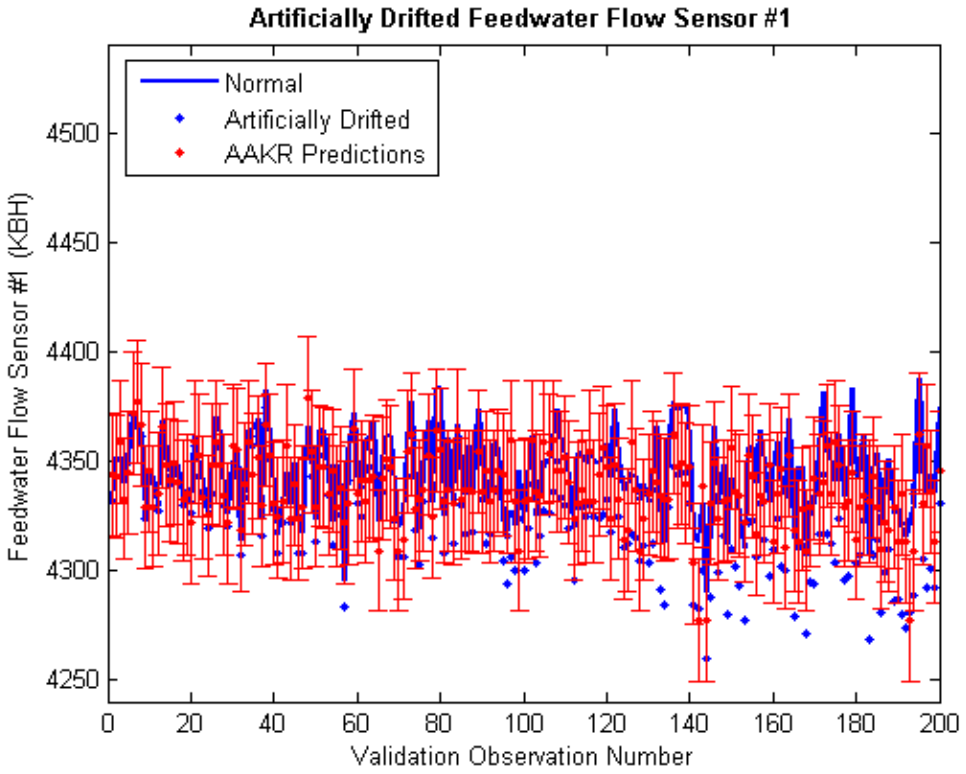


Fig. 6-18. AAKR predictions and for artificially drifted feedwater flow sensor.

Notice that even though the AAKR model is supplied faulted data it is able to predict the normal operating values with a high degree of accuracy.

To quantifiably identify the drift, the AAKR predictions are subtracted from the drifted data providing a residual signal. The residual is then denoised and combined with the bootstrap uncertainty estimates, providing a CI for the residual. The error CI is then compared to the predefined threshold of ±1% to detect the drift. The results of applying EULM are presented in Fig. 6-19. The upper graph in Fig. 6-19 shows the denoised residual of the prediction and actual sensor value along with its uncertainty. The lower graph shows the output of the fault detection algorithm. Notice that the drift is consistently detected by the 105th observation. Since the drift is linear, the drift is identified when the sensor has degraded by approximately 0.5% its nominal value. This is necessary because when the interval crosses the tolerance, one no longer is 95% confident that the sensor has not drifted beyond its calibration tolerance. Spurious alarms could be reduced through filtering or alarm logic. It could be argued that using a filter to reduce the variance is permissible because the drift is considered a bias with very slow dynamics.

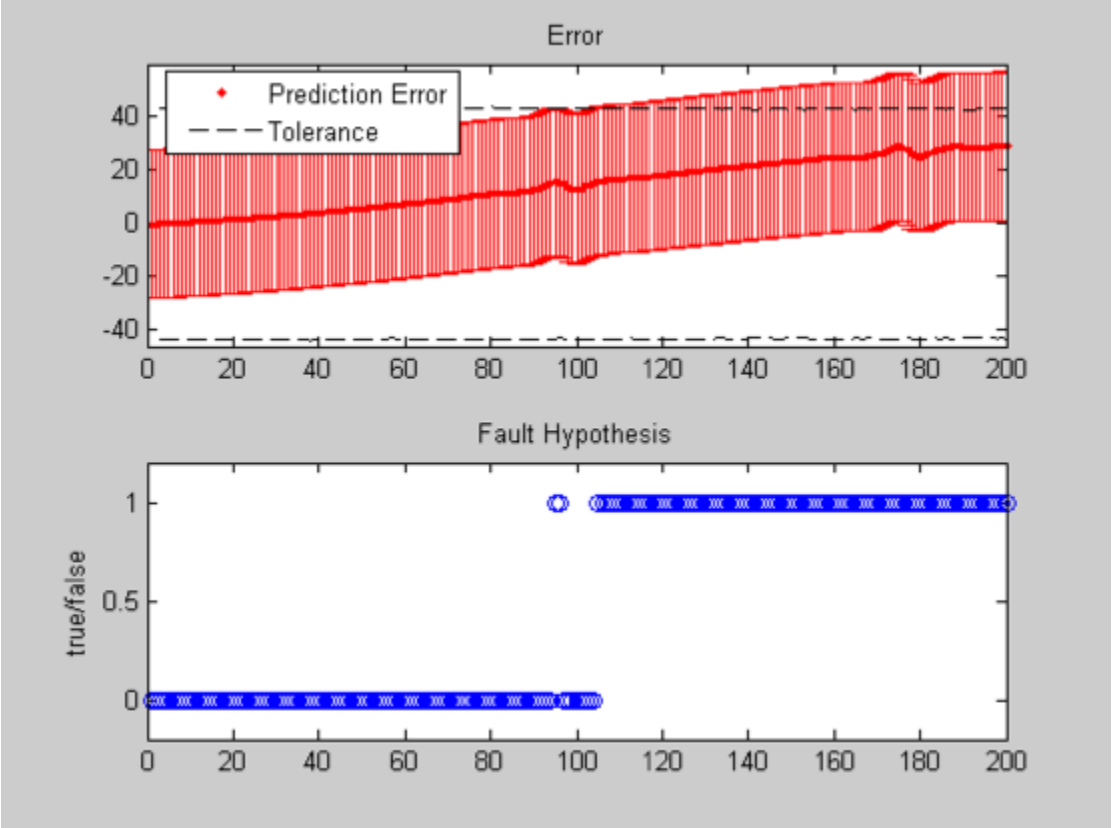


Fig. 6-19. Results of EULM fault detection with AAKR and 1% artificial drift.

6.6 Confidence Intervals vs Prediction Intervals

One consideration that deserves additional discussion is whether the confidence intervals (CIs) or prediction intervals (PIs) should be used in OLM for calibration extension. Experience shows that many sensors contain a significant amount of noise that escalates the PI so that the residual uncertainty exceeds the drift allowance, even when no drift is present. In turn, the smaller CIs can only be applied when the true, noise-free, value is known. To obtain a noise-free estimate of the sensor drift, the residual must be filtered. A usual concern with filtering process signals is that it may remove the process dynamics; however, this concern does not apply to residuals in which the drift dynamics are very slow. Thus, a CI applied to a filtered residual can be employed by OLM uncertainty analyses.

6.6.1 CI vs PI case study

The CI and PI theory was explained in Sect. 3.1.2. This section will investigate the suitability of the PI and CI for sensor calibration monitoring uncertainty quantification. Historically, the PI was considered more appropriate for OLM analyses because it provides the accuracy with which the desired response can be predicted and not just the accuracy of the model itself [Rasmussen 2003]. However, with repeated studies using actual nuclear power plant data, it was found that in some cases the sensor noise can inflate the PI (which includes the noise variance term) to an amount that makes the residual exceed the threshold, even when no drift is present. Figure 6-20 presents the nuclear feedwater flow data used in this study. However, to be thorough, additional studies were conducted using steam flow, steam generator level, turbine pressure, and steam pressure sensor data with similar results.

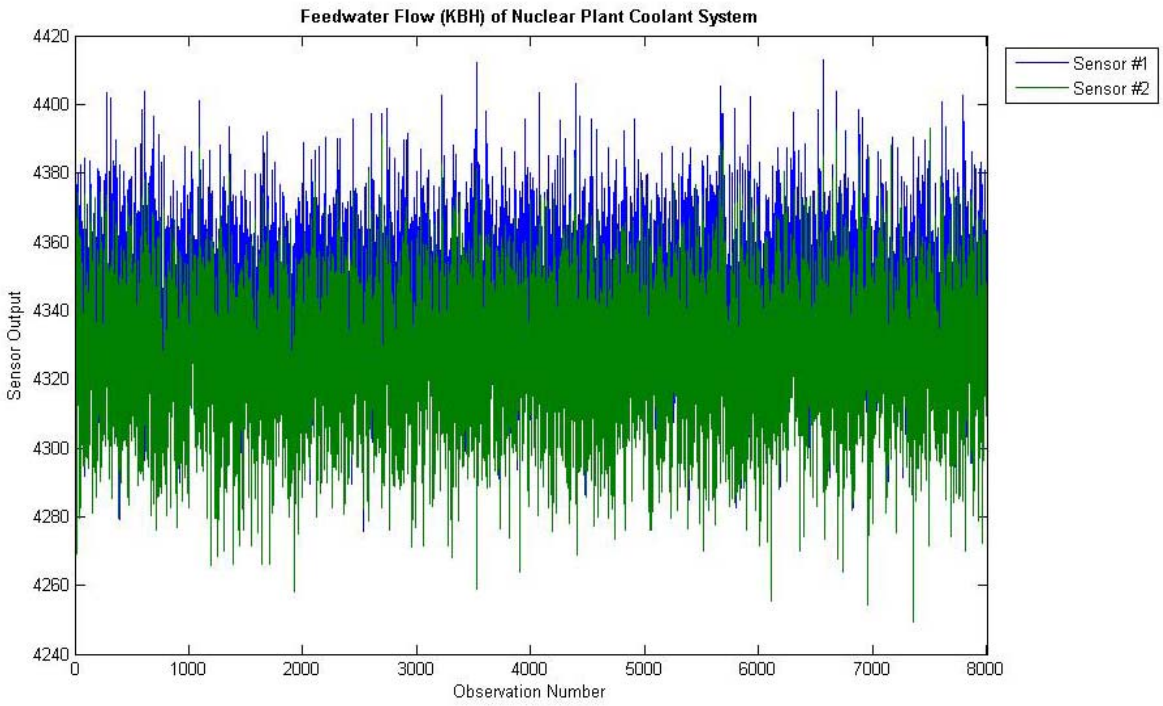
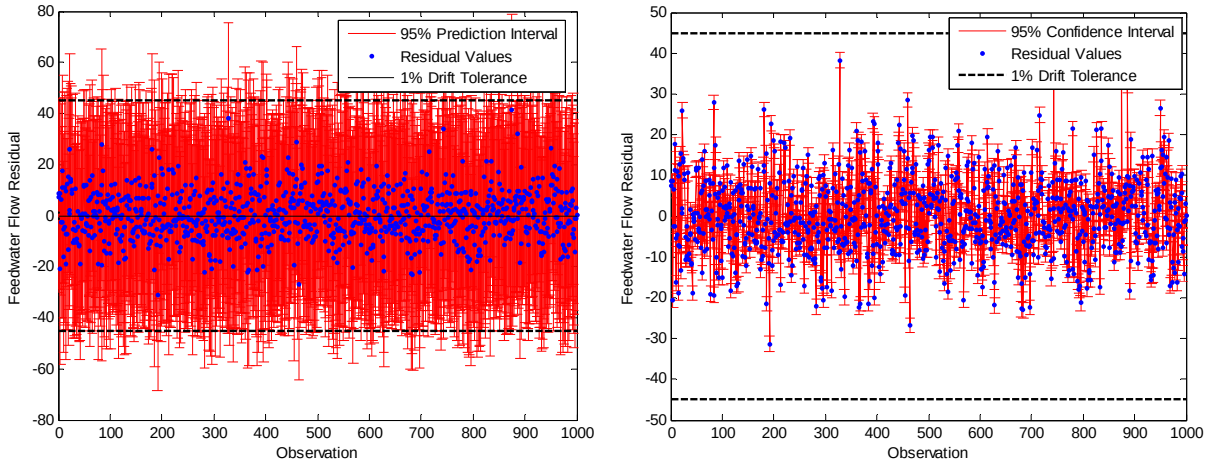


Fig. 6-20. Feedwater flow data.

Figure 6-21(a) illustrates the large PIs produced when noisy sensors are modeled. The figure shows a residual computed by subtracting measured feedwater flow values from estimates made by an auto-associative kernel regression model. The 95% prediction interval for the residual was calculated using Eq. (3.1.2.5). In this equation, the variance was estimated using an analytic technique, the bias was computed using the bias-variance decomposition equations, and the noise variance was estimated by employing kernel smoothing [Wand 1995]. Other techniques for sensor noise prediction include wavelet denoising, ICA filtering, and median filtering. The large prediction interval causes the residual to reach the assumed 1% drift tolerance, even though the sensor has not drifted. Since repeated trials showed this same scenario for many types of sensors, it can be concluded that PIs may not be feasible for OLM uncertainty interval quantification.



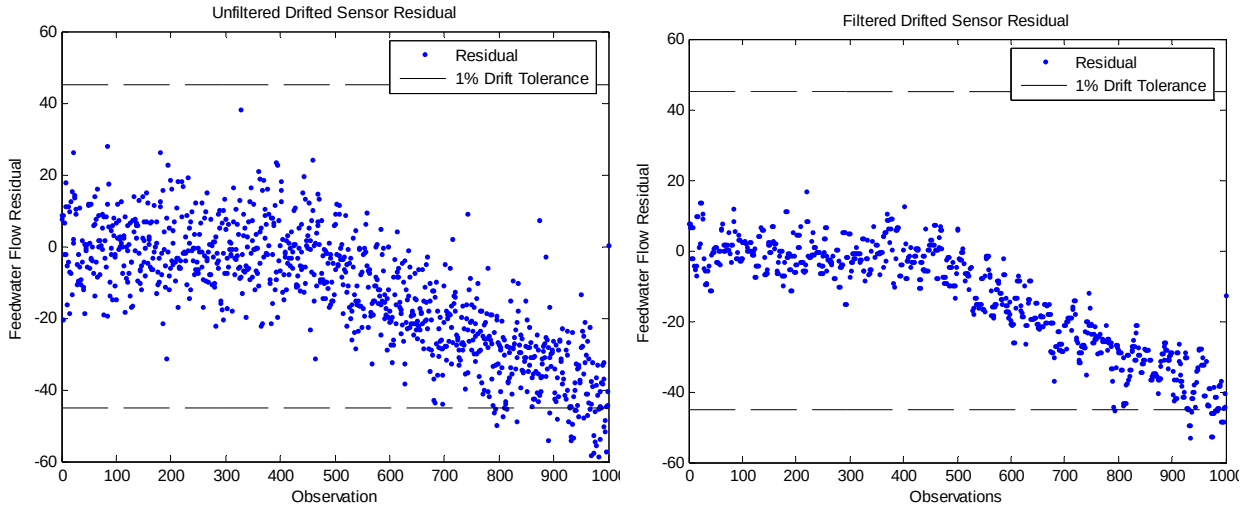
(a) (b)

Fig. 6-21. Residual with (a) 95% prediction interval and (b) 95% confidence interval.

In contrast, CIs usually give acceptable OLM uncertainties. Figure 6-21(b) shows a 95% CI applied to the same feedwater flow residual as used in the previous example. The CI was calculated using Eq. (3.1.2.6). The figure illustrates that the smaller CI does not violate the 1% drift limit. However, there must be justification for using the CI. This needed justification comes from further examination of the CI and its application to the residual.

Again, a 95% PI implies that the *measured value* is covered 95% of the time, whereas a 95% confidence interval implies that the *true value* is covered 95% of the time. As previously discussed, the CI is desired for OLM uncertainty quantification because its values are lower and typically fall below the drift limit. However, the CI assumes that the true value is known and cannot be applied if the assumption is not met. If the system is sampled and a denoising technique is applied, then a good estimate of the noise free variable can be obtained. This would remove the noise term from the uncertainty equation, and the PI would essentially collapse to the CI.

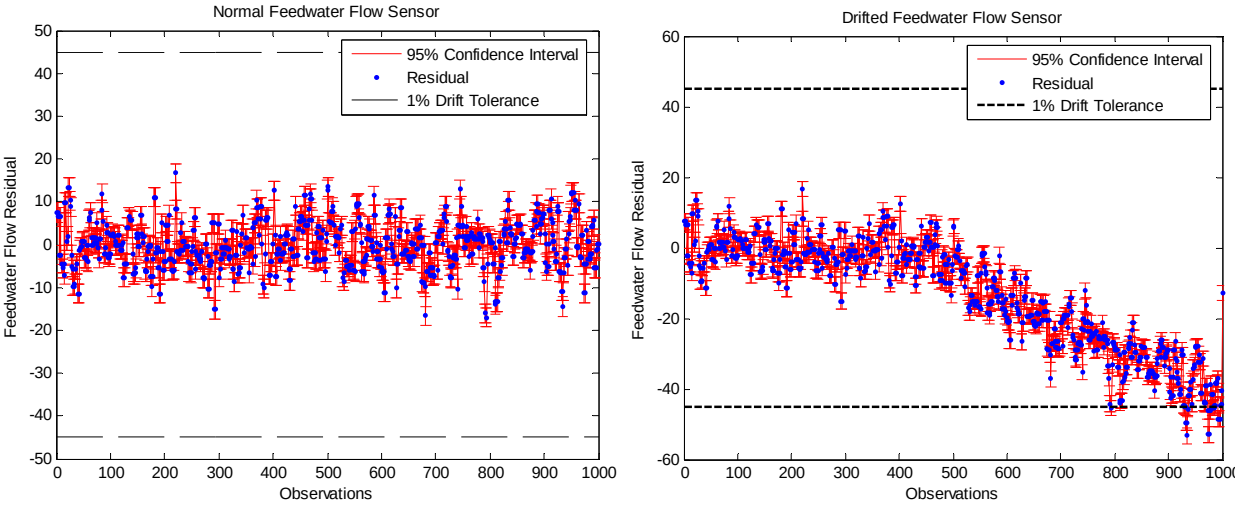
If filtering is used, the bandwidth must be selected so that the process dynamics are retained. To do this, the system must be sampled faster than the process dynamics and the filter set appropriately. However, the residual can be filtered without these requirements. This residual can be filtered because the process dynamics are in both the variable and estimate, and have been cancelled out when forming the residual. The only dynamics in the residual are the drift dynamics. Because drift dynamics are very slow, a lower sample rate is permissible. Figure 6-22 compares the filtered and unfiltered residuals from the same feedwater flow sensor that has been injected with an artificial drift. The drift is still discernable even when filtered. Thus, the residual can be filtered without losing drift dynamics. The filtered residual can be interpreted as the *true value* of the residual, because none of the dynamics have been lost and only the noise removed. This fact means that a CI can appropriately be applied to the filtered residual, because the true value is assumed to be known.



(a) (b)

Fig. 6-22. Drifted feedwater flow residual (a) without filtering and (b) with filtering.

Figure 6-23 compares the filtered residuals of the nondrifting and drifting feedwater flow sensor with 95% confidence intervals.



(a) (b)

Fig. 6-23. Filtered residual with 95% confidence interval of a (a) normally operating feedwater flow sensor, and (b) drifted feedwater flow sensor.

Figure 6-23 illustrates that a CI applied to a filtered residual is capable of detecting drift, but does not exceed the drift limit when no drift is present. Because drift detection is the primary goal of an OLM sensor calibration monitoring system, this result supports the use of CIs in the uncertainty analysis. These results assume the bias and variance were properly quantified.

6.6.2 Conclusion

This chapter has described two common representations of OLM model predictive uncertainty: the confidence interval (CI) and prediction interval (PI). A PI provides the uncertainty for each individual model's prediction, whereas a CI gives the uncertainty for the model's expected prediction. Because the PI is an interval for the value of a single new measurement from the process, it includes the noise that is inherent in the estimates and the uncertainty of the new measurement. This means that the PI is always larger than the CI. These larger PIs can be larger than the drift limit, even when the sensor has exhibited no drift. The smaller CIs generally fall below the drift limit, but can only be used if supported by a theoretical basis. By considering the functionality and theory of the OLM residual, it was determined that the CI could be correctly applied when the residual was filtered and no longer contained the sensor noise. A residual can be filtered without fear of losing the process dynamics. Additionally, it is assumed that since the drift is slow moving, only the noise will be removed when a residual is filtered, providing an accurate estimate of the residual's "true" value. With an estimate of the "true" residual, a CI can appropriately be applied and is considered suitable for OLM.

6.7 Synthesis of Results

This chapter has illustrated several useful features of current uncertainty estimation techniques. A concise summary of these features is listed below.

1. Analytic Methodologies
   (a) When all assumptions are known to be satisfied, they produce uncertainty estimates that are in direct agreement with their theoretical values.
   (b) When all of the assumptions are not known to be satisfied, then the overall uncertainty estimate should change from its actual values. However, this change from the theoretical has been shown to be small when compared to nuclear plant data, whose operation is often about a steady nominal value.
   (c) The analytic methodologies produce uncertainty estimates that are very near asymptotic values of the Monte Carlo methodologies.
2. Monte Carlo Methodologies
   (a) When all assumptions are known to be satisfied, the performance of the direct bootstrap sampling and LHS are nearly equivalent.
   (b) The convergence characteristics of the uncertainty of the estimated variance for direct bootstrap and LHS are nearly equivalent. This indicates that the increased computational complexity of LHS is not necessary to produce reliable and accurate uncertainty estimates.
3. Fault Detectability for Nuclear Plant Data
   (a) The noise levels in the feedwater flow and steam flow sensors prevent the application of the PI.
   (b) When the CI is applied to denoised prediction residuals, the uncertainties are generally within the ±1% calibration tolerance. Therefore, the use of the CI for fault detection is applicable for instrument calibration monitoring systems.

Although this section demonstrated OLM's ability to verify a sensor's calibration status, several factors may limit the application and usefulness of the predictive modeling methods. Some of the concerns with these methods will be discussed in the following chapter, which provides a brief summary of the report's contents and then attempts to address any potential issues surrounding OLM. Additionally, Volume 3 of this NUREG series will investigate several modeling assumptions and present the effects of not meeting those assumptions.

7. CONCLUSIONS AND ADDITIONAL CONCERNS

7.1 Summary

The goal of this report was to present an evaluation of the most current OLM techniques that have been investigated for use in extending the required manual calibration interval of safety critical instrument channels. The report presented several empirical prediction models, namely AANN, AAKR, and AAMSET, and investigated their use for OLM. The report provided a theoretical basis for the analysis of the model predictive uncertainty. The assumptions in both the Monte Carlo and analytically based uncertainty predictions strategies were also evaluated. The report used both simulated and real data to compare both uncertainty methods. The comparison showed that when all assumptions were satisfied, both the analytical and Monte Carlo methods produce predictions in agreement with their theoretical values.

7.2 Concerns

The usage of OLM systems requires that the assumptions inherent in the systems will always be met, or in cases when they are not met, either the impact is known to be insignificant, or proper safeguards are in place. This implies that the OLM method and uncertainty analysis cannot be treated simply as a "black box." Rather, there must be a comprehensive knowledge of the system's theoretical foundations so that in the cases in which there is an anomaly or an assumption is no longer valid, proper actions can be taken before plant safety is compromised. Engineering judgment must often be exercised when making decisions regarding the actions that must be taken in response to potential problems with an OLM system. In the following sections, some of the basic assumptions surrounding the different functional tasks in the OLM process are discussed. Volume 3 of this series will investigate several of the assumptions in more detail, quantify the results of not meeting the assumptions, and provide methods to identify cases in which the assumptions are not met.

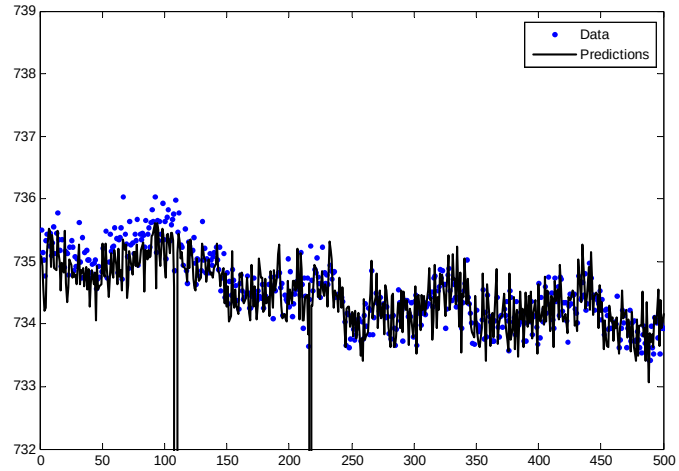
7.2.1 Data assumptions

The results presented in this report were constructed using several underlying assumptions. The main assumptions were that the data used to build the model was error-free and representative of the system and that it covered the entire future operating range. This last assumption is extremely significant because the described empirical models (in fact all nonlinear empirical models) are unable to extrapolate outside the range in which they have been trained. If the input data fall outside of the training space, there can be no confidence in the model's prediction for that input. This fact may not pose great concern for nuclear utilities, because they generally operate at 100% power in a standard configuration. However, seasonal variations, abnormal operations, and equipment repair or failure can cause a change in the operating conditions and a change in the measured plant parameter correlations. Therefore, operational processes must be in place to ensure that the model is operating within the training region. The PEANO system developed by the Halden Reactor Group in Norway has a reliability assessment module that verifies that the model is operating within the training region. This module uses distance measures between the current state and trained states to determine if the operating condition is inside the trained region. A similar module should be integrated into all safety-critical OLM systems, and procedures should be in place that defines how to proceed when the model's training data no longer cover the operating space. For a more thorough discussion of data and model quality, the reader is referred to Chaps. 4 and 6 in the first volume of this series and Chaps. 4 and 5 of Volume 3.

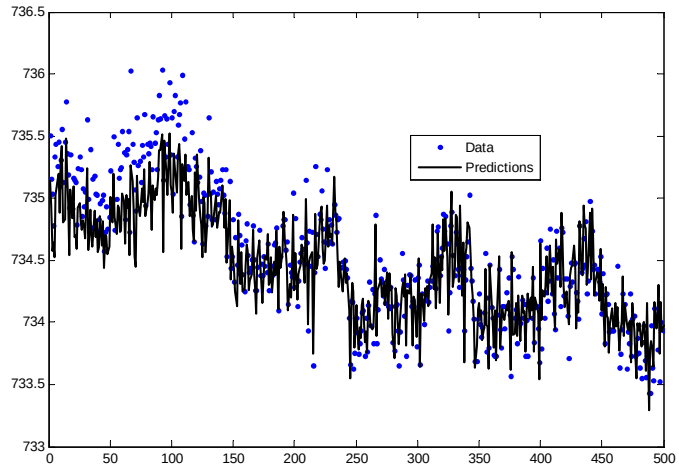
7.2.2 Uncertainty assumptions

The assumptions in the uncertainty analysis play a large role in determining the adequacy of OLM. Since the *ADVOLM* and *MAVD* drift limits (discussed in detail in Volume 1 of this series) are calculated from a model's uncertainty, it is necessary that the uncertainty analysis be correct and the underlying assumptions fully understood and met.

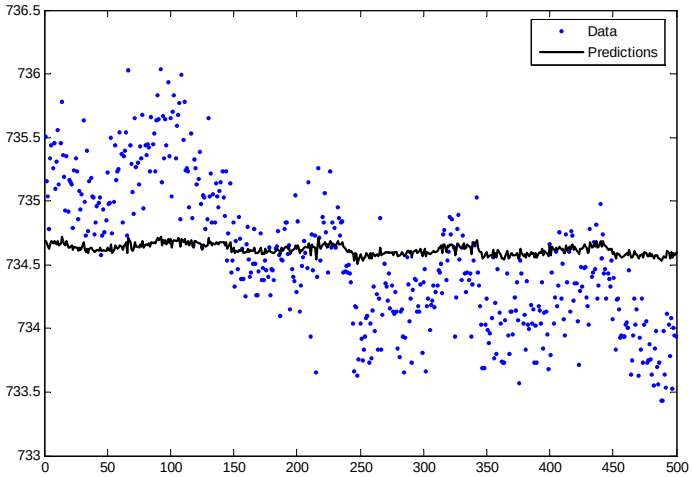
All of the uncertainty analyses presented in this report operate under the assumption that the underlying modeling technique is appropriate for the data set. For example, the analyses assume that a linear model is not being applied to nonlinearly related data. All of the modeling techniques presented in this report are considered Universal Function Approximators [Hornick 1989] and are appropriate for OLM applications, so this assumption is of little concern unless the model complexity is overly constrained through reduction of hidden neurons in the AANN or reduction of the kernel bandwidth in AAKR or AAMSET. If a bandwidth is set too high or there are too few neurons, a model oversmoothes the data, and the features become obscured with an increase in bias. Conversely, if a bandwidth is set too low or too many neurons are included in the architecture, a model will fit the noise in the data rather than the underlying functional relationship, resulting in an increased variance. The key is that a model should have the proper bandwidth or number of neurons to allow its complexity to match the complexity of the data. Figure 7-1 illustrates this principle. The figure shows that when a kernel bandwidth of 0.05 is used, the bandwidth is too small, and the model fits each data point, including the noise, perfectly. When a bandwidth of 0.5 is used, the predictions smooth some of the noise. However, when a bandwidth of 50 is used, the model oversmoothes the data resulting in the prediction being similar to the mean value of the data.



(a)



(b)



(c)

Fig. 7-1. AAKR model of turbine pressure sensor with a (a) kernel bandwidth of 0.05, (b) kernel bandwidth of 0.5, and (c) kernel bandwidth of 5.

Another concern with the uncertainty analyses is how to obtain the final OLM uncertainty estimate. The Monte Carlo analyses yields uncertainty estimates for each observation in the validation set. However, a single uncertainty value is needed to set the drift limits. The proposed method uses the 95th percentile largest uncertainty estimate selected from the point-wise estimates and then uses a t-value for a 95% confidence when converting the standard error to a CI or PI. This results in a conservative, 95/95 estimate of the single uncertainty value.

7.2.3 Bootstrap assumptions

The direct bootstrap method for uncertainty quantification requires fewer assumptions and is consistent with the random character of the predictors in many empirical modeling applications. The only assumptions inherent in the bootstrap method are that the noise is truly random and the errors are independent. These assumptions are usually by nuclear data sets unless special situations arise.

An additional concern when using the bootstrap is that enough simulations have been performed to gain an accurate prediction of the model's true statistics. There is no rule dictating how many simulations are necessary to achieve adequate results. Generally, for most OLM applications, the number of simulations should be in the thousands to ensure that the samples have been thoroughly exercised and that the resulting uncertainty estimate converges to the true uncertainty value. It is possible to check that the number of simulations is adequate by repeating the bootstrap process several times and verifying that the results have stabilized. This may prove to be extremely time-consuming, but it is necessary to produce confidence in the results and is performed off-line when time is not as much of a concern.

7.2.4 LHS assumptions

LHS requires more assumptions than the direct bootstrap procedure. Along with the assumptions present in bootstrap sampling, LHS also assumes that each variable's probability distribution can be adequately modeled. Fortunately, with a large number of observations, the data tend to be normally distributed, in accordance with the Central Limit Theorem. However, it is not wise to simply assume normally distributed data. Rather, statistical tests must be performed to predict and verify the data's distribution. This task becomes even more complicated with multivariate data when joint probability distributions must be determined. The difficulty of determining the data's distribution may outweigh the benefit of needing fewer simulations to estimate the model's variance. However, if LHS is employed, it is important to realize that the probability distributions of the data set must be properly modeled in order for the technique to be valid.

7.2.5 Assumptions in the analytical uncertainty methods

The major assumption of the analytic equations for the AANN is that the variance-covariance matrix **S** may be represented by:

$$\mathbf{S} = \left[\mathbf{F}^{\mathrm{T}} \mathbf{F} \right]^{-1} . \tag{7.2.5.1}$$

From Eq. (7.2.5.1), it is assumed that it is possible to derive an estimate of the relative change in a model's predictions vs its parameters for a given input and that the prediction is valid. There is also the assumption that the model inputs are noise-free. Finally, the AANN equations rely on the assumption that a first-order Taylor series expansion of a model about its parameters is sufficient to describe the sensitivity. This requires that the sensitivity can be linearized around the operating point, which is valid because it performs this calculation for each sample.

If these assumptions are not completely met, the uncertainty estimates inherently increase. For example, if the actual operating condition moves away from the trained conditions, the uncertainty estimates increase. These effects will be explored in more detail and reported in Volume 3 of this series.

In addition to the method-specific assumptions, there are several major assumptions in the PI and CI equations themselves. First, it is assumed that the errors are normally distributed, giving a proper estimate of a 95% uncertainty. This is a reasonable expectation and can be validated through statistical analysis. Additionally, the PI and CI equations assume that the bias-variance decomposition is a valid method for estimating uncertainty and that the methods for estimating bias, discussed in Sect. 4.3, are valid. These decompositions have been reported throughout the literature and are considered valid.

7.2.6 Assumptions in estimating the bias

The methods presented in this report to estimate the bias component of the uncertainty all assume that the model's bias is constant for each variable and may be approximated by its expected value. Although this assumption is generally true, for areas of high nonlinear correlations, which are characterized by sharp curvature in the data relationships, the bias will increase significantly and may invalidate this assumption. However, nuclear process data are generally collected when a plant is operating at 100% power and does not exhibit much curvature. Another assumption in the bias estimation is that "true" or noise-free signals can be determined. A "true" signal is assumed to have all of the independent noise removed, while retaining the common process noise. These noise-free signals are needed to estimate the bias term in the uncertainty analysis, as well as to quantify the noise, and its distribution, present in each sensor input if LHS is used in the analyses. Thus, the following section discusses the assumptions surrounding several of the denoising methods.

7.2.7 Wavelet denoising assumptions

The major assumptions of wavelet denoising are that wavelet decomposition and reconstruction have the ability to estimate the "true" signal. These techniques are becoming widespread, and there is a growing base of information that supports this assumption. The other assumptions are associated with scoring each denoising parameter set's performance. The scoring assumptions include the following: (1) the noise is normally distributed and uncorrelated with itself (white); (2) the noise removed is uncorrelated with other process variables; and (3) the smoothing has maximally reduced the variance while satisfying the previous assumptions. These assumptions are somewhat appropriate for OLM, with the exception of normally distributed noise. This assumption comes from the central limit theorem, which states that the summation of noise sources with any distribution tends toward normal as the number of sources tends toward infinity. If there are dominating sources, this might not hold. Chapter 7 of Volume 3 of this NUREG series investigates this in detail.

7.2.8 Independent component analysis assumptions

The major assumptions of the ICA algorithm are that the data are time-invariant. Because plants usually operate at nearly 100% power, the assumption of time-invariant data is usually met. Obviously, when the plant undergoes a transient, the data become nonstationary, and the ICA method fails. Generally, transient data are often excluded from OLM estimations. However, if transient data are being used for some reason, such as a reliability analysis during startup or shutdown, the fact that this assumption is violated must be realized and a different denoising method employed. Additionally, the ICA algorithm is usually applied to data off-line to estimate noise variances. The on-line implementation has not been commercially applied to OLM, and the stationarity of the signal should not be a concern.

The second restriction in using ICA denoising is that it cannot separate more than one Gaussian component. If two Gaussian components are mixed, they cannot be separated in the future. This is not a concern because the underlying process parameter rarely has a Gaussian distribution that needs to be

separated from the Gaussian noise components. Recall that noise sources are commonly assumed to be Gaussian because when several noise sources are added, they tend toward Gaussian; this does not imply that the original sources were Gaussian. Chapter 7 of Volume 3 of this NUREG series investigates denoising in detail and shows that the method of denoising has little effect on the predictive uncertainty; therefore, complex filtering techniques such as ICA would probably not be used.

7.2.9 Filtering

The Analysis and Measurement Services Corporation (AMS) has implemented filtering algorithms in their OLM implementation at Sizewell B in the United Kingdom. Their argument is that drift is a very slow phenomena and can be estimated by filtering to reduce the variance component of uncertainty. This is valid if the data are sampled so that the filter bandwidth is above the dynamics of the process variable. Stated another way; the filter bandwidth should be set to remove the random variations, not the process variations. More information on the Sizewell B implementation is available through a report published by EPRI in 2006 [EPRI and British Energy Group (2006)].

The concept of filtering to remove the variance has led some to believe that the only uncertainty component of interest should be the bias portion and that the variance portion can be neglected. Because the variance portion of the uncertainty is usually many times larger than the bias portion, this argument could have a significant impact on the drift limits. However, the analytical equations used to estimate the variance portion of the uncertainty are also sensitive to the assumptions inherent in the technique as discussed in Sect. 7.2.4 and provide a fail-safe mechanism to validate the assumptions. The bootstrap technique is not sensitive to changing assumptions because the uncertainty is estimated offline. This is another reason why additional techniques to validate assumptions, such as operating condition coverage, are necessary, especially for the Monte Carlo techniques. For these reasons, the total uncertainty (bias plus variance) should be estimated and used to set the drift limits.

7.2.10 Detectability

Although all of the performance metrics discussed in Chap. 2 are important, auto and cross-sensitivity are of particular concern. If a model's cross or auto-sensitivity is much larger than 0, then the model will be affected by drift and may be unsuitable for OLM. Even with small sensitivity values, a model will still be unable to detect drift as accurately as it should, and the drift limits for the OLM system may need to be modified to reflect this fact. For instance, if a model has an auto-sensitivity of 0.2 and the drift limit is at 1%, the drift limit may need to be changed to 0.8% to account for the fact that the model will exhibit a 2% drift if any of the sensors in it have drifted. EULM fault detectability (discussed in Chap. 2) provides a method for modifying the drift limits accordingly. Still, it is clear that this detectability issue warrants exploration so that the proper safeguards can be in place. Volume 3 in this series will further examine both model sensitivity and detectability and their impact on OLM.

7.2.11 Setting drift limits

Establishment of the ADVOLM and MAVD is a plant-specific task. However, certain considerations must be taken into account when calculating these limits. These include

- monitoring interval and drift rate assumption,
- single-point monitoring penalty,
- predictive uncertainty, and
- sensitivity.

The concern with monitoring interval arises when plants choose not to perform OLM on a continual basis. The sensor calibration uncertainties, which are used to compute the MAVD, are unique to each plant and/or sensor. Although these terms are rather straightforward in that they are stipulated in the

plant's set point calculation documentation, the monitoring frequency still must be considered when using them to establish the drift limit. The sensor drift (which is a term included in the set point calculation) is the specified allowance for drift at the interval between refueling outages. If OLM is not conducted continuously, then the full sensor drift value cannot be used because there is still the possibility that the sensor has drifted between surveillances. For instance, if OLM calibration assessment is performed quarterly, then the sensor drift value is scaled from its original value, which gives the drift allowance during the entire fuel cycle, to a value that subtracts off the small drift allowance for the time between surveillances. To perform this subtraction, assumptions about the drift rate must be made. Most commonly, the drift is assumed to be linear and the allowance that accounts for the interval between surveillances is calculated using this assumption.

Additionally, the single-point monitoring penalty, which is also used to compute the MAVD, must be identified for each individual sensor. EPRI [2000] conducted a large drift study. The results of this study supplied generic single-point monitoring penalties for a variety of sensor types and models. However, if a sensor is not included in the EPRI drift study, the single-point monitoring penalty must be calculated using historical calibration data. If these data are unavailable or if too few data are available, the plant must show that the generic EPRI penalty for a similar instrument is applicable.

Finally, the predictive uncertainty term must be quantified for each specific model used at each plant. The uncertainty calculation methods provided in this report are not necessarily the only correct way to compute a model's predictive uncertainty. Although these methods are known to be correct and result in safe systems when the assumptions are met, many other uncertainty derivations are found in literature and could potentially be applied to OLM. NIST Technical Note 1297 [1994] outlines many other methods for computing uncertainty. As long as a plant has a sound technical basis behind its uncertainty calculations, the calculations should be valid and appropriate for determining the ADVOLM. However, as discussed in Sect. 2 and reiterated in Sect. 7.10, the MAVD must also consider a model's sensitivity (or robustness).

Determining the ADVOLM is also a problematical task. The ADVOLM is a conservative limit set slightly below the MAVD that when exceeded declares that the sensor is in need of calibration at the next outage (rather than declaring the sensor inoperable.) If the ADVOLM is set too high, there is a greater risk of an instrument exceeding the ADVOLM and being declared inoperable before it has a chance to be calibrated. Conversely, if the ADVOLM is set too low, more instruments will exceed it, and unnecessary calibrations will be performed. Historical data can be analyzed to determine a value for the ADVOLM based on an instrument's normal operation.

7.3 Conclusions

This report explains the role of the AANN, AAKR, and AAMSET model in an OLM system. In addition to presenting the theory behind these models, the report attempts to clarify each model's merits and perceived shortcomings in its application to OLM. Several methods for quantifying the different modeling technique's uncertainty are derived. Finally, each of the models, along with the various methods of uncertainty analysis, are applied to both simulated and actual nuclear plant data. The applications demonstrated that under normal operating conditions the modeling techniques and uncertainty analyses presented in this report yield results suitable for OLM. The calculated uncertainty values appear to accurately reflect the true predictive uncertainty of the specific models, with good agreement between the analytical and Monte Carlo based estimates. Furthermore, these calculated uncertainty values were usually small enough to show that OLM could be used to detect sensor drift before limits are reached. To elaborate, if the technique's predictive uncertainty had been too great, the acceptable band would have been reduced to the point that it no longer covers the sensor's normal operating region, and OLM would have been ineffective, as it would declare normally operating sensors as needing calibration. However, the uncertainty values for both the real and simulated data were reasonable and fell within a normal sensor's acceptable band. Although this application produced promising results, it failed to validate many of the assumptions inherent in OLM modeling. The majority

of these assumptions are addressed in the previous section. Many of these assumptions are concealed until an abnormality occurs in the OLM system. Thus, Volume 3 of this NUREG series, entitled *Limiting Case Studies*, will apply the modeling and uncertainty analysis techniques to a wide variety of plant data sets and model development routines to explore the modeling assumptions and limitations.

Volume 3 will be the final report in this NUREG series and summarizes seven case studies investigating the effects of model development and assumptions on model performance. Two case studies concern the effect of not meeting model assumptions: evaluating query data outside the training region and training with faulty data. Recommendations are given for identifying and correcting the problems caused by not meeting these important data assumptions. The third and fourth case studies investigate the effects of high noise levels on model performance and compare different methods of data denoising, respectively. The remaining three case studies examine different features of model development by comparing vector selection methods, different numbers of memory vectors, and robust distance measures. Methodologies to determine the appropriate model development parameters for each of these cases are outlined. Finally, a section is included that highlights the special considerations needed for redundant-sensor model architectures.

8. REFERENCES

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski (1985), "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, **9**, 147–169 (1985).

Atkeson, C. G., A. W. Moore, and S. Schaal (1997), "Locally Weighted Learning," *Artificial Intelligence Review*, **11**, 11–73 (1997).

Bello, M. G. (1992), "Enhanced Training Algorithms, and Integrated Training/Architecture Selection for Multilayer Perceptron Networks," *IEEE Trans. on Neural Net.*, **3**, 864–875 (1992).

Bickford, R. L., et al. (2000), *MSET Signal Validation System for Space Shuttle Main Engine,* Final Report, NASA Contract NAS8-98027, August 2000.

Bickford, R. L., C. Meyer, and V. Lee (2001), "Online Signal Validation for Assured Data Quality," *Proc. of the 2001 Instrument Society of America*, 2001.

Bickford, R., R. E. Holzworth, R. D. Griebenow, and A. Hussey (2003), "An Advanced Equipment Condition Monitoring System for Power Plants," *Trans. Am. Nucl. Soc.*, *New Orleans, LA, Nov. 16–20, 2003*.

Chapelle C., V. Vapnik, O. Bousquet, and S. Mukherjee (2002), "Choosing Multiple Parameters for Support Vector Machines," *Machine Learning*, 2002.

Chauvin, Y. (1989), "Towards a Connectionist Model of Symbolic Emergence," *Proc. of the 11th Annual Conference of the Cognitive Science Society*, pp. 580–587 (1989).

Cleveland, W. S., and C. Loader (1994a), *Computational Methods for Local Regression*, Technical Report 11, AT&T Bell Laboratories, Statistics Department, Murray Hill, NJ, 1994.

Cleveland, W. S., and C. Loader (1994b), *Smoothing by Local Regression: Principles and Methods*, Technical Report 95.3, AT&T Bell Laboratories, Statistics Department, Murray Hill, NJ, 1994.

Coleman, H. W., and W. G. Steele (1998), *Experimentation and Uncertainty Analysis for Engineers*, J. Wiley & Sons, New York, 1998.

Coppock, S. and T. Hopton (2003), "Advanced Software Detects Temperature Indicator Failure," *Trans. Amer. Nuc. Soc.*, *New Orleans, LA, November 2003*.

Cotrell, G. W., P. Munro, and D. Zipser (1987), "Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming," *Proc. of the 9th Annual Conference of the Cognitive Science Society*, pp. 461–473 (1987).

Chryssolouris, G., M. Lee, and A. Ramsey (1996), "Confidence Interval Prediction for Neural Network Models," *IEEE Transactions on Neural Networks*, **7**(1), 229–232 (1996).

Demuth, H., and M. Beale (2004), *Neural Network Toolbox for Use with MATLAB: User's Guide*, Version 7, The MathWorks Inc., 2004.

Demuth, H., M. Beale, and M. Hagan (2005), *Neural Network Toolbox For Use with MATLAB: User's Guide Version 4*, The MathWorks Inc.: 2005.

Ding, J. (2004), "Independent Component Analysis for Sensor Validation," Ph.D. Dissertation, University of Tennessee, Knoxville, TN, August 2004.

Dong, D., and T. J. McAvoy (1994), "Sensor Data Analysis Using Auto-associative Neural Nets," *Proc. of the World Congress on Neural Networks*, **1**, 161–166 (June 5–9, 1994).

Dong, D. and T. J. McAvoy (1996), "Nonlinear Principal Component Analysis—Based on Principal Curves and Neural Networks," *Computers and Chemical Engineering*, **20**, 65–78 (1996).

Dybowski, R. (1997), *Assigning Confidence Intervals to Neural Network Predictions*, Technical Report 2, March, Summary of presentation at the Neural Computing Applications Forum (NCAF), London, January 1997.

Efron, B., and R. J. Tibshirani (1993), *An Introduction to the Bootstrap*, Chapman and Hall, New York, 1993.

EPRI (1993), TR-103436-V1, *Instrument Calibration and Monitoring Program, Volume 1: Basis for the Method*, EPRI, Palo Alto, CA, 1993.

EPRI (1995), *Monte Carlo Simulation and Uncertainty Analysis of the Instrument Calibration and Monitoring Program*, WO3785-02, EPRI, Palo Alto, CA, 1995.

EPRI (2000), *On-Line Monitoring of Instrument Channel Performance*, Topical Report 104965-R1, Final Report 100604, EPRI, Palo Alto, CA, 2000.

EPRI (2004), *On-Line Monitoring of Instrument Channel Performance Volume 3: Applications to Nuclear Power Plant Technical Specification Instrumentation*, Final Report 1007930, EPRI, Palo Alto, CA, 2004.

EPRI and British Energy Group PLC (2006), *On-Line Monitoring for Calibration Interval Extension of Safety Related Instruments: Vol. 1*, EPRI, Palo Alto, CA, and British Energy Group PLC, Suffolk, UK, 2006.

Fan, J., and I. Gijbels (1996), *Local Polynomial Modeling and Its Applications*, Chapman & Hall/CRC, New York, 1996.

Fantoni, P., S. Figedy, and A. Racz (1998), *A Neuro-Fuzzy Model Applied to Full Range Signal Validation of PWR Nuclear Power Plant Data*, FLINS-98, Antwerpen, Belgium.

Fantoni, P. (1999), "On-Line Calibration Monitoring of Process Instrumentation in Power Plants," EPRI Plant Maintenance Conference, Atlanta, Georgia, June 21, 1999.

Fantoni, P., M. Hoffmann, B. Rasmussen, J. W. Hines, and A. Kirschner (2002), "The Use of Nonlinear Partial Least Square Methods for On-Line Process Monitoring as an Alternative to Artificial Neural Networks," *5th International Conference on Fuzzy Logic and Intelligent Technologies in Nuclear Science (FLINS)*, Gent, Belgium, Sept. 16–18, 2002.

Feller, W. (1945), "The Fundamental Limit Theorems in Probability," *Bull. Amer. Math. Soc.,* **51**, 800–832 (1945).

Fukunaga, K., and W. Koontz (1970), "Application of Karhunen-Loeve Expansion to Feature Selection and Ordering," *IEEE Transactions on Computing*, **C-19**, 311–320 (1970).

Gribok, A., J. W. Hines, A. Urmanov, and R. E. Uhrig (2002), "Heuristic, Systematic, and Regularization for Process Monitoring," Special issue of the *International Journal of Intelligent Systems on Intelligent Systems for Process Monitoring*, Wiley Publishers, 2002.

Gribok, A. V., A. M. Urmanov, and J. W. Hines (2004), "Uncertainty Analysis of Memory Based Sensor Validation Techniques," *Real-Time Systems*, **27**(1) (May 2004).

Gross, K. C., S. Wegerich, and R. M. Singer (1998), "New Artificial Intelligence Technique Detects Instrument Faults Early," *Power Magazine*, **42**(6), 89–95, McGraw-Hill (1998).

Gross, K. C., R. M. Singer, S. W. Wegerich, J. P. Herzog, R. VanAlstine, and F. Bockhurst (2001), "Application of a Model-Based Fault Detection System to Nuclear Plant Signals," *Proc. Intelligent System Application to Power Systems (ISAP '97), Seoul, Korea, July 6–10, 2001.*

Gross, K. C., V. Bhardwaj, and R. L. Bickford (2002a), "Proactive Detection of Software Aging Mechanisms in Performance-Critical Computers," *Proc. 27th Annual IEEE/NASA Software Engineering Symp.*, *Greenbelt, MD, Dec. 4–6, 2002*.

Gross, K. C., W. Lu, and D. Huang (2002b), "Time-Series Investigation of Anomalous CRC Error Patterns in Fibre Channel Arbitrated Loops," *Proc. 2002 IEEE Int'l Conf. on Machine Learning and Applications (ICMLA)*, *Las Vegas, NV, June 2002*.

Gross, K. C., and W. Lu (2002c), "Early Detection of Signal and Process Anomalies in Enterprise Computing Systems," *Proc. 2002 IEEE Int'l Conf. on Machine Learning and Applications (ICMLA)*, *Las Vegas, NV, June 2002*.

Gross, K., V. Bhardwaj, and R. Bickford (2002d), "Proactive Detection of Software Aging Mechanisms in Performance Critical Computers," *27th Annual IEEE/NASA Software Engineering Symposium*, *Greenbelt, MD, December 4–6, 2002*.

Hagan, M. T., and M. B. Menhaj (1994), "Training Feed Forward Network with the Marquardt Algorithm," *IEEE Trans. on Neural Net*., **5**(6), 989–993 (1994).

Hashemian, H. M., *On-Line Testing of Calibration of Process Instrumentation Channels in Nuclear Power Plants, Phase II Final Report*, NUREG/CR-6343 (1995), U.S. Nuclear Regulatory Commission, Washington, D.C., November 1995.

Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, Macmillan Publishing Company, Inc., Englewood Cliffs, NJ, 1994.

Helton, J. C. (2004), "Sampling-Based Methods for Uncertainty and Sensitivity Analysis," *Proc. 4th Int. Conf. on Sensitivity Analysis of Model Output, Santa Fe, NM, March 8–11, 2004*.

Heskes, T. (1998), "Bias/Variance Decomposition for Likelihood-Based Estimators," *Neural Computation*, **10**(6) (August 1998).

Hines, J. W., A. Gribok, A. Urmanov, and R. E. Uhrig (2002), "Heuristic, Systematic, and Informational Regularization for Process Monitoring," *International Journal of Intelligent Systems on Intelligent Systems for Process Monitoring*, Wiley Publishers, **17(**8), 723–750 (2002).

Hines, J. W., and A. Usynin (2005), "MSET Performance Optimization Through Regularization," *Nuclear Engineering and Technology*, **38**(2) (April 2005).

Hines, J. W., and D. Garvey (2005), "An Auto-associative Empirical Modeling Toolbox for On-Line Monitoring," *The 18th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management (COMADEM)*, *Cranfield, Bedfordshire, United Kingdom, August 31–September 2, 2005*.

Hines, J. W., and A. Usynin (2004), "On-Line Monitoring Robustness Measures and Comparisons," *International Atomic Energy Agency Technical Meeting on "Increasing Instrument Calibration Interval Through On-Line Calibration Technology,"* OECD Halden Reactor Project, Halden, Norway, September 27–29, 2004.

Hines, J. W., and D. Garvey (2006), "Development and Application of Fault Detectability Performance Metrics for Instrument Calibration Verification and Anomaly Detection," to appear in the *Journal of Pattern Recognition Research*.

Hodouin, D., J. F. MacGregor, M. Hou, and M. Franklin (1993). "Multivariate Statistical Analysis of Mineral Processing Plant Data," *CIM Bulletin, Mineral Processing*, **86**(975), 23–34 (1993).

Hoerl, A. E., and R. W. Kennard (1970), "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, **12**(1), 55–82 (1970).

Hopfield, J. J. (1982), "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the American Academy of Science, U.S.A.*, **79**, 2554–2558 (1982).

Hornick, K., M. Stinchcombe, and H. White (1989), "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, **2**, 359–366 (1989).

Hotelling, H., (1933), "Analysis of a Complex of Statistical Variables into Principal Components," *The Journal of Educational Psychology*, 498–520 (1933).

Hyvarinen, A., J. Karhunen, and E. Oja (2001), *Independent Component Analysis*, John Wiley and Sons, New York, 2001.

International Organization for Standardization (1993), *Guide to the Expression of Uncertainty in Measurement*, ISO, Geneva, 1993.

Kosko, B. (1988), "Bidirectional Associative Memories," *IEEE Trans. Systems Man. Cybernet.*, **18**, 49–60 (1988).

Kramer, M. A. (1991), "Nonlinear Principal Component Analysis Using Auto-Associative Neural Networks," *American Institute of Chemical Engineers Journal*, **37**, 233–243 (1991).

Kramer, M. A. (1992), "Auto-Associative Neural Networks," *Computers & Chemical Engineering*, **16**(4), 313–328 (1992).

Lippmann, R. P. (1987), "An Introduction to Computing with Neural Networks," *IEEE ASSP Magazine*, **35**, 4–22 (1987).

Miron, A. (2001), "A Wavelet Approach for Development and Application of a Stochastic Parameter Simulation System," Ph.D. Dissertation, University of Cincinnati, Cincinnati, OH, 2001.

Miron, A., and J. Christenson (2003), "The Stochastic Parameter Simulation System: A Wavelet-Based Methodology for "Perfect" Signal Reconstruction," *ANS Topical Meeting in Mathematics and Computations, Gatlinburg, TN, April 6–11, 2003*.

Misiti, M., Y. Misiti, G. Oppenheim, and J.-M. Poggi (2004), *Wavelet Toolbox for Use with MATLAB: User's Guide Version 3*, The MathWorks Inc., 2004.

Morgan, M. G., and M. Henrion (1990)*, Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*, Cambridge University Press, Cambridge, UK, 1990.

Mott, Y., and R. W. King (1987), *Pattern Recognition Software for Plant Surveillance*, U.S. DOE Report.

Nadaraya, E. A. (1964), "On Estimating Regression," *Theory of Probability and Its Applications*, **10**, 186–190 (1964).

NIST Technical Note 1297 (1994), B. N. Taylor and C. E. Kuyatt, *Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results*, NIST, 1994.

NRC Project No. 669 (2000), "Safety Evaluation by the Office of Nuclear Reactor Regulation: Application of On-Line Performance Monitoring to Extend Calibration Intervals of Instrument Channel Calibrations Required by the Technical Specifications," EPRI Topical Report (TR) 104965, *On-Line Monitoring of Instrument Channel Performance*, U.S. Nuclear Regulatory Commission: Washington, D.C., July 2000.

Rasmussen, B., E. Davis, and J. W. Hines (2002), "Monte Carlo Analysis and Evaluation of the Instrumentation and Calibration Monitoring Program," *Proc. Maintenance and Reliability Conference (MARCON 2002), Knoxville, TN, May 7–10*.

Rasmussen, B. (2003), "Prediction Interval Estimation Techniques for Empirical Modeling Strategies and Their Application to Signal Validation Tasks," Ph.D. Dissertation, The University of Tennessee, Knoxville, TN, 2003.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986), "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, Vol. 1, MIT, Cambridge, MA (1986).

Schimek, M. G. (2000), *Smoothing and Regression*, Wiley Series in Probability and Statistics, John Wiley and Sons, New York, 2000.

Scott, D. W. (1992), *Multivariate Density Estimation*, Wiley, New York, 1992.

Singer, R. M., K. C. Gross, J. P. Herzog, R. W. King, and S. Wegerich (2001), "Model-Based Nuclear Power Plant Monitoring and Fault Detection: Theoretical Foundations," *Proc. Intelligent System Application to Power Systems (ISAP '97*), Seoul, Korea, July 6–10, 2001.

Snyman, J. A. (2005), *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*, Springer Publishing.

Stone, M. (1974), "Cross-Validatory Choice and Assessment of Statistical Predictions," *Journal of the Royal Statistical Society*, B, **36**(1), 111–147 (1974).

Tamhane, A. C., and D. D. Dunlop (2000), *Statistics and Data Analysis from Elementary to Intermediate*, Prentice Hall, Upper Saddle River, NJ, 2000.

te Braake, H. A. B., H. J. L. V. van Can, and H. B. Verbruggen (1998), "Semi-Mechanistic Modeling of Chemical Processes with Neural Networks," *Engineering Applications of Artificial Intelligence*, **11**(4), 507–515 (1998).

Thompson, M. L., and M. A. Kramer (1994), "Modeling Chemical Processes Using Prior Knowledge and Neural Networks," *AIChE Journal*, **4**(8), 1328–1340 (1994).

Tibshirani, R. (1996), "A Comparison of Some Error Estimates for Neural Network Models," *Neural Computation*, **8**, 152–163 (1996).

Tsoukalas, L., and R. Uhrig (1997), *Fuzzy and Neural Approaches in Engineering*, John Wiley and Sons, New York, 1997.

Uhrig, R. E., J. W. Hines, D. J. Wrest, C. J. Black, and X. Xu (1996), *Instrument Surveillance and Calibration Verification System*. Report Prepared by the University of Tennessee for Sandia National Laboratories, Contract No. AQ-6982, 1996.

Upadhyaya, B. R. (1985), "Sensor Failure Detection and Estimation," *Nuclear Safety*.

Upadhyaya, B. R., and K. Holbert (1989), *Development and Testing of an Integrated Signal Validation System for Nuclear Power Plants*, DOE Contract DE-AC02-86NE37959.

Upadhyaya, B. R., and E. Eryurek (1992), "Application of Neural Networks for Sensor Validation and Plant Monitoring," *Nuclear Technology*, **97**, 170–176 (February 1992).

Wand and Jones (1995), *Kernel Smoothing*, Monographs on Statistics and Applied Probability, Chapman & Hall, London, 1995.

Watson, G. S. (1964), "Smooth Regression Analysis," *The Indian Journal of Statistics*, Series A, **26**, 359–372 (1964).

Wegerich, S., R. Singer, J. Herzog, and A. Wilks (2001), "Challenges Facing Equipment Condition Monitoring Systems," *Proc. Maintenance and Reliability Conference*, *Gatlinburg, TN*.

Wegerich, S. (2002), "Performance Comparison of Variable Selection and Grouping Algorithms," SmartSignal Corp. Technical Report for Funded Research Project: "MSET Variable Selection Optimization," August 2002.

Wrest, D., J. W. Hines, and R. E. Uhrig (1996), "Instrument Surveillance and Calibration Verification Through Plant Wide Monitoring Using Auto-associative Neural Networks," published in the *Proc. of The 1996 American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies*, *University Park, PA, May 6–9, 1996*.

Wilson, J. A., and L. F. M. Zorsetto (1997), "A Generalized Approach to Process State Estimation Using Hybrid Artificial Neural Network Mechanistic Models," *Computers & Chemical Engineering*, **21**(9), 951–963 (1997).

Wolpert, D. H., and W. G. Macready (1995), *No Free Lunch Theorems for Search*, Technical Report SFI-TR-95-02-010 (Santa Fe Institute).

Wyss, G. D., and K. H. Jorgensen (1998), *A User's Guide to LHS: Sandia's Latin Hypercube Sampling Software, Risk Assessment and Systems Modeling Department*, Sandia National Laboratories, Albuquerque, NM, 1998.

Zavaljevski, N., A. Miron, C. Yu, and T. Y. C. Wei (2004), "A Study of On-line Monitoring Uncertainty Based on Latin Hypercube Sampling and Wavelet De-Noising," *Fourth ANS Topical Meeting on Nuclear Plant Instrumentation*, *Controls and Human-Machine Interface Technologies*, *Columbus, OH, September 2004*.

ANSI/ISA–67.04.01(2000), "Setpoints for Nuclear Safety-Related Instrumentation," ISA, Research Triangle Park, NC, 2000.

ISA-RP67.04.02 (2000), *Methodologies for the Determination of Setpoints for Nuclear Safety-Related Instrumentation*," ISA, Research Triangle Park, NC (2000).

Page intentionally blank